НІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри
_____ Аліна САВЧЕНКО
«___»_____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВР

ЗА ОСВІТНЬО-ПРОФЕСІЙНОЮ ПРОГРАМОЮ

«ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ПРОЕКТУВАННЯ»

Тема: «JSON parser для роботи з базою даних»

Виконавець: Богдан ГРИЦАК

Керівник: к.т.н. Олег ЗУДОВ

Нормоконтролер: к.т.н., доцент Олена ТОЛСТІКОВА

КИЇВ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних наук та технологій*
Кафедра *комп'ютерних інформаційних технологій*
Спеціальність *122 «Комп'ютерні науки»*
Освітньо-професійна програма *«Інформаційні технології проектування»*

ЗАТВЕРДЖУЮ:
завідувач кафедри КІТ
_____Аліна САВЧЕНКО
(підпис)
«___»_____2023 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи
Грицака Богдана Ярославовича
(ПІБ випускника)

1. Тема роботи: «JSON parser для роботи з базою даних» затверджена наказом ректора № 623/ст від 01.05.2023р.

2. Термін виконання роботи: з 15 травня 2023 року по 25 червня 2023 року.

3. Вихідні дані до роботи: готовий додаток «JSON parser для роботи з базою даних»

4. Зміст пояснювальної записки: 1. Основи структури та принципів JSON формату. 2. Вибір та обгрунтування інструментів проектування 3. Створення інтерфейсу та алгоритму програми. 4. Підключення до бази даних, стилізація, реалізація.

5. Перелік обов'язкового ілюстративного матеріалу: 1. JSON. 2. JSON parser. 3. Вибір середовища розробки для програмування на мові Python. 4. Вибрані бібліотеки Python для розробки проекту. 5. Встановлення вибраних бібліотек. 6. Створення інтерфейсу JSON parsera. 7. Програмування JSON parsera 8. Демонстрація роботи JSON parsera.

6. Календарний план-графік

№ з/п	Завдання	Термін виконання	Підпис керівника
1	Огляд та аналіз предметної області. Написання 1 розділу, представлення керівнику.	15.05.2023-20.05.2023	
2	Вибір та опис використаних технологій. Написання 2 розділу, представлення керівнику.	21.05.2023-27.05.2023	
3	Написання 3 та 4 розділу, представлення керівнику.	28.08.2023-04.06.2023	
4	Загальне редагування та друк пояснювальної записки.	05.06.2023-08.06.2023	
5	Проходження нормоконтролю, перепліт пояснювальної записки.	09.06.2023-15.06.2023	
6	Розробка тексту доповіді. Оформлення графічного матеріалу для презентації	16.06.2023-18.06.2023	

7. Дата видачі завдання 15.05.2023р.

Керівник кваліфікаційної роботи _____ Олег ЗУДОВ
(підпис керівника)

Завдання прийняв до виконання _____ Богдан ГРИЦАК
(підпис випускника)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи на тему: «JSON parser для роботи з базою даних» містить: 60 сторінок, 29 рисунків, 11 інформаційних джерел.

Об’єкт дослідження – парсинг JSON формату.

Предмет дослідження – JSON parser для роботи з базою даних.

Мета кваліфікаційної роботи – розробити програмне забезпечення, яке зможе зчитувати дані у форматі JSON, перетворювати їх у зручний для подальшої обробки вигляд та надавати користувачеві необхідні інструменти для маніпуляції цими даними.

Методи дослідження – аналіз літературних джерел та наукових робіт, дослідження принципів та структури JSON формату, вивчення існуючих бібліотек та інструментів для роботи з JSON у мові програмування Python, а також процес розробки та тестування власного JSON парсера. Крім того, буде проведено порівняльний аналіз з іншими аналогічними рішеннями на ринку з метою оцінки ефективності та функціональності розробленого інструменту.

Результати кваліфікаційної роботи з розробки JSON parsera можна використовувати для аналізу, парсингу, документації, роботи зі складними JSON рядками.

Для розробки JSON parsera мовою Python було знайдено та використано наступні ефективні програмні засоби: бібліотека PyQt5, бібліоиека Qt, бібліотека json, бібліотека pyjq, бібліотека psycopg2.

PYTHON РОЗРОБКА, JSON, JSON PARSER, QT DESIGNER, DESKTOP APPLICATION, CSS СТИЛІЗАЦІЯ

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

БД	-	База даних
AJAX	-	Технологія звернення до серверу без перезавантаження сторінки
NaN	-	Not a Number
IDE	-	Інтегроване середовище розробки
СУБД	-	Система управління базами даних

ВСТУП

У сучасному світі обробка та аналіз даних стають все більш важливими завданнями, що передують успішному розвитку технологій та бізнесу. Один із найпоширеніших форматів для обміну та зберігання даних - JSON, знайшов широке застосування в сфері програмування та баз даних. JSON має просту та зручну структуру, що дозволяє легко обробляти дані, та його популярність зростає з кожним днем.

Однак, для ефективної роботи з даними у форматі JSON потрібен потужний інструмент, який зможе зчитувати та обробляти дані відповідно до потреб користувача. Таким інструментом є JSON-parser, програмне забезпечення, яке дозволяє перетворювати дані у форматі JSON у зручний та читабельний вигляд для подальшої обробки.

Актуальність теми кваліфікаційної роботи «JSON parser для роботи з базою даних» полягає у зростаючій потребі розробників у зручних та ефективних інструментах для обробки даних у форматі JSON.

Об’єктом дослідження є парсинг JSON формату.

Предмет дослідження – це JSON parser для роботи з базою даних.

Мета кваліфікаційної роботи – розробка програмного забезпечення, яке зможе зчитувати дані у форматі JSON, перетворювати їх у зручний для подальшої обробки вигляд та надавати користувачеві необхідні інструменти для маніпуляції цими даними.

У відповідності з метою роботи визначено основні **завдання дослідження**:

− дослідження JSON формату, ретельне ознайомлення із його структурою та особливостями, його синтаксисом та можливостями. Вивчення основних правил та концепцій, таких як об'єкти, масиви, ключі та значення.

− аналіз доступних інструментів та бібліотек для обробки JSON даних у мові програмування Python. Вивчення їх можливостей, функцій та обмежень. Порівняння різних бібліотек та вибір оптимального варіанту для реалізації JSON-parsera.

− аналіз різних середовищ розробки для мови програмування Python та вибір оптимального середовища, що відповідає вимогам дипломної роботи. Врахування таких факторів, як зручність використання, наявність необхідних інструментів та підтримка розробки.

− проектування архітектури та інтерфейсу JSON-parsera. Реалізація алгоритмів для зчитування та обробки даних у форматі JSON, забезпечення ефективності та надійності роботи.

− проведення тестування розробленого JSON-parsera для перевірки його ефективності, швидкодії та точності обробки даних. Порівняння з існуючими аналогами та оцінка переваг та недоліків розробленого JSON-parsera.

Для досягнення мети та виконання поставлених завдань використано наступні методи: порівняльний, обробка літературних джерел, аналітичний.

Наукова новизна полягає в розробці нового підходу до парсингу JSON даних, який включає в себе нові алгоритми обробки, відкритість та безпечність коду, оптимізацію продуктивності та впровадження нових функціональних можливостей

Практичне значення отриманих результатів полягає в можливості ефективної обробки та аналізу JSON даних. Розроблений JSON-parser може бути використаний розробниками для обробки та перетворення даних у форматі JSON в їх проектах, особливо тих, які пов'язані з роботою з БД.

РОЗДІЛ 1
ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Основи структури та принципів JSON формату

JSON (JavaScript Object Notation) - це текстовий формат обміну даними, який широко використовується в сучасній веб-розробці. JSON формат був створений для того, щоб узгодити формат даних між різними мовами програмування, що дозволяє легко передавати та обробляти дані між клієнтським та серверними додатками.

Формат JSON використовується для впорядкованого зберігання даних у процесі їх обміну між веб-браузером (або клієнтською частиною програми) та сервером (або між різними серверами). Більше того, завдяки текстовому вигляду рядка дані JSON можна легко передавати через будь-які інші канали обміну інформацією в інтернеті. Для отримання доступу до даних, що містяться в ньому, файл .json повинен бути перетворений на об'єкт JavaScript, для чого в цій мові є відповідні методи перетворення.

Якщо описувати принцип дії цього інструменту на реальному приклад, то, наприклад, є певний веб-додаток, який зберігає та обробляє дані своїх користувачів: текстовий редактор, поштовий сервіс або будь-що. Без JSON ці відомості досить незручно зберігати: вони будуть недоступні при заході з іншого браузера/пристрою (у разі зберігання в пам'яті браузера) або їх взагалі потрібно вписувати вручну (у разі зберігання прямо всередині документа).

Для вирішення цієї проблеми і був винайдений стандарт JSON, який помітно спрощує та прискорює взаємну передачу даних між клієнтом (інтернет-браузер) та сервером сайту. Інакше кажучи, це спосіб запису об'єктів у JavaScript. Він оптимально взаємодіє з AJAX, разом вони

забезпечують асинхронне завантаження даних у фоновому режимі. Така функція дозволяє сайтам та веб-програми оновлювати інформацію без обов'язкового перезавантаження сторінок. Крім того, за допомогою JSON користувачам доступний запит даних із стороннього домену.

JSON використовується для представлення даних у вигляді пар "ключ-значення" (key-value pairs). Ключі є рядками, а значення можуть бути рядками, числами, логічними значеннями (true/false), масивами чи об'єктами. Об'єкти в JSON представляються у фігурних дужках і містять набір пар "ключ-значення", а масиви представляються у квадратних дужках і містять послідовність значень.

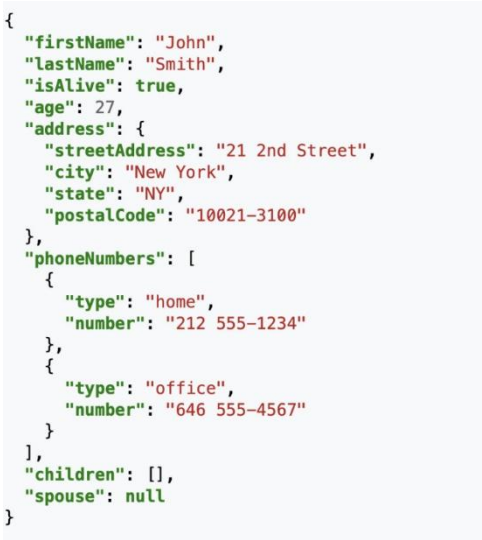

Рис. 1.1. Приклад JSON формату

JSON має декілька переваг порівняно з іншими форматами обміну даними, зокрема:

- Маленький розмір файлів
- Незалежність від мов програмування
- Широке використання в сучасній веб-розробці
- Легка читабельність

Головним аналогом JSON є XML (Extensible Markup Language), його було створено раніше ніж JSON. XML виникло в 1998 році і стало широко використовуваним форматом для обміну даними та зберігання

структурованої інформації. JSON, з іншого боку, був розроблений пізніше - в 2001 році. Він був призначений для спрощення обміну даними у веб-додатках та програмуванні, зосереджуючись на простоті та лаконічності. Обидва формати - XML і JSON - продовжують використовуватися в сучасних програмах, проте JSON став більш популярним в розробці веб-додатків та API-інтерфейсів завдяки своїй простоті та зручності в розумінні. За рахунок своєї простоти в порівнянні з XML, формат JSON може бути більш придатним для серіалізації складних структур.

Рис. 1.2. Порівняння XML та JSON

Якщо говорити про вебзастосунки, у такому ключі JSON доречний у задачах обміну даними як між браузером і сервером (AJAX), так і між самими серверами (програмні HTTP-інтерфейси). Формат JSON так само добре підходить для зберігання складних динамічних структур у реляційних базах даних або файловому кеші.

1.2. Типи даних які підтримуються в JSON

JSON має дуже обширний список можливостей та підтримуваних типів даних, що допомагає розробникам ефективно обмінюватися та зберігати різноманітні дані, такі як числа, рядки, булеві значення, масиви, об'єкти, а також спеціальні типи даних, такі як null та вбудовані значення. Це робить JSON універсальним форматом для передачі та зберігання даних в різних програмних середовищах та системах. Крім того, JSON також дозволяє

використовувати власні користувацькі типи даних та розширювати його функціональність за допомогою розширених схем та метаданих.

JSON підтримує наступні основні типи даних:

1. Рядки (Strings): Рядки в JSON представляються у вигляді послідовностей символів, заключених у подвійні лапки. Наприклад, "Hello, World!". Рядки можуть містити будь-які символи Unicode, і їх можна складати разом для утворення більш складних рядків.
2. Числа (Numbers): JSON підтримує цілі числа (наприклад, 42) та числа з плаваючою крапкою (наприклад, 3.14). Вони використовуються для представлення числових значень у JSON об'єктах.
3. Булеві значення (Booleans): Булеві значення в JSON можуть бути true (істина) або false (хиба). Вони використовуються для представлення логічних станів, таких як стан увімкнено/вимкнено.
4. Масиви (Arrays): Масиви в JSON є упорядкованими списками значень, розділеними комами та заключеними у квадратні дужки. Наприклад, [1, 2, 3]. Масиви можуть містити будь-які типи даних, включаючи рядки, числа, булеві значення, об'єкти та вкладені масиви.
5. Об'єкти (Objects): Об'єкти в JSON представляють собою неупорядковані колекції пар "ключ-значення". Вони використовують фігурні дужки для визначення початку та кінця об'єкта, а кожна пара "ключ-значення" в об'єкті має вигляд "ключ: значення". Наприклад, {"name": "John", "age": 30}. Ключі є рядками, а значення можуть бути будь-якого типу даних, включаючи рядки, числа, булеві значення, масиви та вкладені об'єкти.

Крім основних типів даних, JSON також підтримує спеціальні значення, наприклад null, що означає відсутність значення.

JSON також підтримує додаткові типи даних, які розширюють його функціональність. Ось кілька прикладів таких типів даних, які підтримуються JSON:

1. Дата і час: JSON дозволяє використовувати рядкове представлення дати і часу у певному форматі, наприклад, у форматі ISO 8601 ("2023-06-03T10:30:00Z").
2. Двійкові дані: JSON може представляти двійкові дані у вигляді рядка шістнадцяткових цифр або базе64-кодування.
3. Вбудовані значення: JSON підтримує вбудовані значення, такі як Infinity і NaN, які можуть бути використані для представлення спеціальних числових значень.
4. Коментарі: JSON не має вбудованої підтримки коментарів, але коментарі можуть бути додані до JSON-файлів у документації або для зручності розробки. Проте, коментарі не є частиною стандартного синтаксису JSON і не будуть оброблені парсерами JSON.
5. Власні користувацькі типи даних: JSON дозволяє використовувати власні користувацькі типи даних, які можуть бути представлені об'єктами з визначеними властивостями та значеннями.

Загалом, це лише кілька прикладів додаткових типів даних, які підтримуються JSON. Широкі можливості JSON дозволяють розробникам ефективно використовувати його для обміну різноманітних даних в різних сценаріях програмування та зберігання даних [1-2].

1.3. Використання JSON в сфері програмування та баз даних

JSON є незамінним інструментом для передачі даних у програмуванні та базах даних, використовуваних у різних сферах. Веб-розробники використовують JSON для передачі різноманітних даних між клієнтом і сервером у веб-додатках, включаючи конфігураційні параметри, результати запитів до бази даних та об'єкти, які представляють сутності додатка. Його легкість у використанні та зчитуваність роблять його популярним форматом для розробників API, які мають необхідність обмінюватись даними між різними системами.

У сфері баз даних JSON також займає своє важливе місце. Деякі бази даних, такі як MongoDB, дозволяють зберігати та працювати з даними у форматі JSON. Це дозволяє зберігати структуровані документи без потреби в реляційній схемі та спрощує роботу з даними зі складною структурою. Крім того, JSON може бути використаний як формат обміну даними між різними базами даних, полегшуючи інтеграцію та обмін даними між різними системами.

Одна з головних переваг JSON в програмуванні - його широка сумісність з багатьма мовами програмування. Більшість сучасних мов мають вбудовану підтримку JSON, що дозволяє зручно працювати з JSON-даними без необхідності використовувати сторонні бібліотеки. Це спрощує обробку та генерацію JSON-даних, забезпечуючи розробникам ефективну роботу незалежно від мови програмування, яку вони використовують.

Крім того, JSON підтримує вкладеність та посилання на інші об'єкти, що робить його особливо потужним при роботі зі складними структурами даних. Це дозволяє представляти зв'язки між об'єктами та колекціями даних, зберігаючи їх в зручному та легкодоступному форматі. Такий підхід є надзвичайно корисним при роботі зі складними об'єктами або при збереженні зв'язків між даними. [3]

1.4. Призначення JSON parser

JSON parser - це програмний засіб або бібліотека, яка використовується для аналізу та обробки даних у форматі JSON. В основному його завдання полягає в трансформації рядка JSON у структуровані дані, які можуть бути легко оброблені та використані в програмі.

Основними функціями та призначенням JSON парсера є:

1. Аналіз та перетворення: JSON парсер дозволяє аналізувати рядок JSON і перетворювати його у внутрішню структуру даних, таку як об'єкти, масиви, рядки, числа тощо. Це дозволяє програмі отримувати доступ

до окремих елементів JSON та використовувати їх для подальшої обробки.

2. Валідація: JSON парсер перевіряє коректність формату JSON даних, виконуючи перевірки на правильність синтаксису та структуру. Це допомагає гарантувати, що дані, які передаються у форматі JSON, є вірними та можуть бути безпечно оброблені.
3. Доступ до даних: JSON парсер надає зручний доступ до окремих елементів даних в JSON форматі. За допомогою JSON парсера програма може отримати значення конкретного поля, перебрати елементи масиву, виконувати умовні перевірки тощо. Це дозволяє ефективно взаємодіяти з даними та виконувати різні операції з ними.
4. Інтеграція та обмін даними: JSON парсери широко використовуються для обміну даними між різними системами та сервісами. Вони дозволяють передавати дані у форматі JSON, що є легким для сприйняття та обробки, і забезпечують сумісність між різними програмними компонентами.

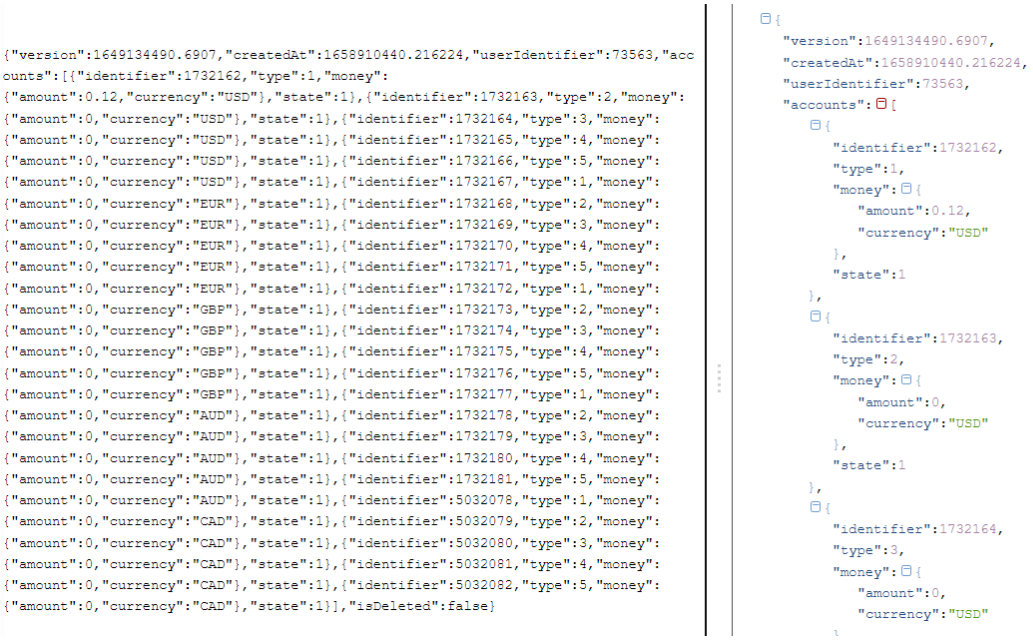

Рис. 1.3. До і після роботи parser

Структурування данних дуже пришвидшує та дозволяє спростити роботу з даними у форматі JSON та забезпечити їх більш ефективне

використання в програмному забезпеченні, що в свою чергу позитивно впливає на швидкість виконання поставлених перед розробником завдань.

1.5. Огляд та порівняння аналогів

1. Json Parser Online

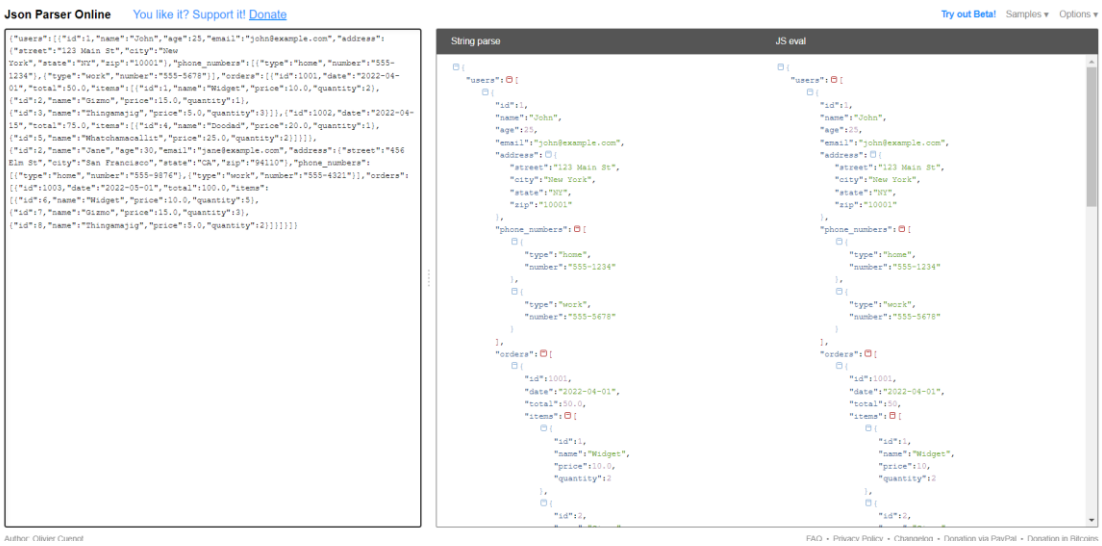

Рис. 1.4. Інтерфейс Json Parser Online

Сайт json.parser.online.fr - це онлайн-інструмент для аналізу та перетворення даних в форматі JSON. Цей інструмент дозволяє користувачам ввести рядок JSON-коду та перетворити його на зрозумілі формати.

При введенні рядка JSON-коду на сайті, користувач отримує структурований результат, який легко зрозуміти. Для того, щоб зрозуміти дані, сайт відображає їх у вигляді дерева, де кореневим елементом є об'єкт JSON. Кожний ключ або значення, відповідні кореневому елементу, представлені в дереві як дочірні елементи.

Окрім цього, сайт підтримує перетворення JSON-даних в різні формати, такі як CSV, XML, YAML та інші. Користувач може також скопіювати результат роботи інструменту в буфер обміну або завантажити його у вигляді файлу на свій комп'ютер.

Серед мінусів можна відмітити:

- Залежність від доступності Інтернету, так як він вимагає постійного підключення до нього для роботи. Відповідно, якщо присутні

проблеми доступу до Інтернету або обмежена швидкість підключення, це може бути проблемою.

- Конфіденційність даних: При використанні онлайн сервісів, особливо якщо передається конфіденційна інформація, є ризик витоку даних. Потрібно мати на увазі, що дані можуть бути доступні власникам або операторам сервісу.
- Використання онлайн сервісу означає залежність від наявності та функціональності цього сервісу. Якщо сервіс перестане працювати або буде недоступний, змога використовувати його пропаде.
- В усіх онлайн парсерах можуть мати обмеження щодо швидкості парсингу або обробки даних. Якщо потрібно обробити великі файли або виконати складні операції з JSON, це може зайняти багато часу.

2. JSON Viewer & Formatter

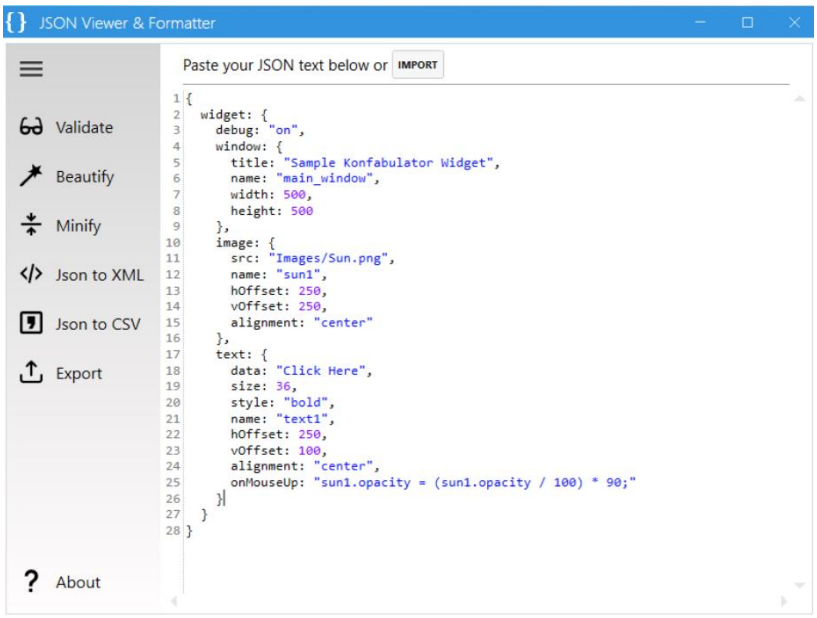

Рис. 1.5. Інтефейс JSON Viewer & Formatter

Json Viewer & Formatter - це додаток, який доступний для завантаження в Microsoft Store. Цей додаток дозволяє користувачам переглядати та форматувати дані у форматі JSON.

Додаток пропонує кілька режимів відображення, включаючи деревовидний та списковий режим. В деревовидному режимі користувач

може розгорнути кожен об'єкт JSON, щоб побачити його вміст. В списковому режимі дані відображаються у вигляді списку, де кожен елемент містить ключ та значення.

Серед мінусів варто відмітити, що розширення в Магазині Microsoft можуть не отримувати часті оновлення або підтримку з боку розробників. Це може призвести до затримок в усуненні помилок або відсутності нових функцій, які можуть бути доступні в інших версіях розширення.

3. JSON formatter

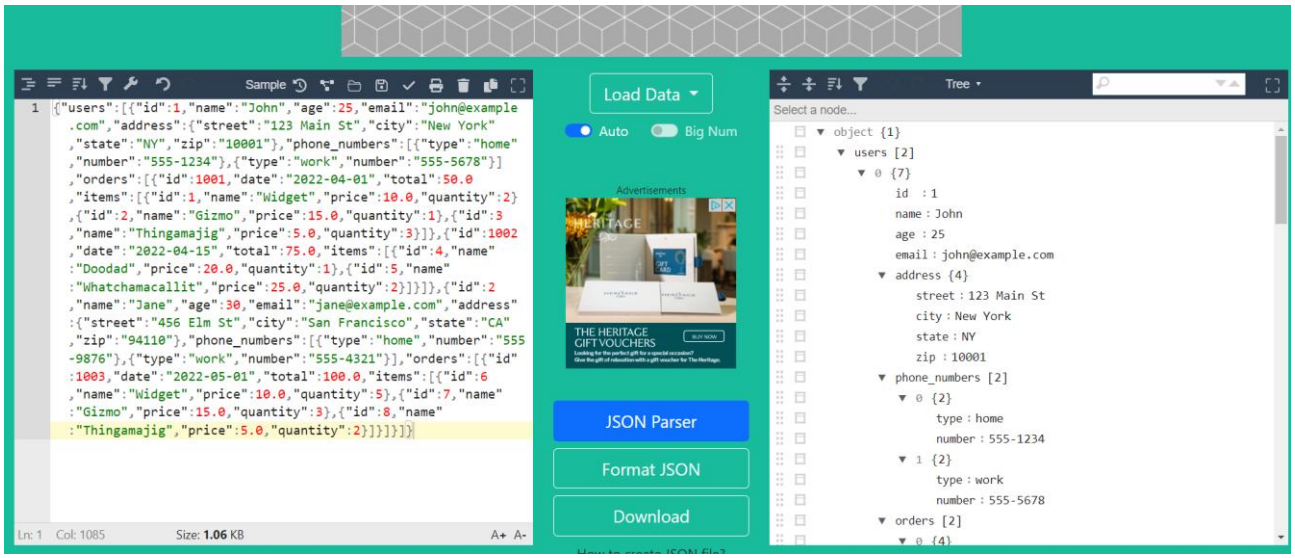

Рис. 1.6. Інтерфейс JSON formatter

Jsonformatter.org це онлайн-інструмент для форматування та перевірки коректності синтаксису даних у форматі JSON. Він дозволяє користувачам зручно переглядати, редагувати та перевіряти свої дані.

Jsonformatter.org також надає користувачам можливість зберігати та завантажувати файли JSON. Це може бути корисним для тих, хто працює з великими обсягами даних.

Як і в випадку з Json Parser Online однією з головних проблем виступає конфіденійність інформації. При використанні онлайн сервісів, особливо якщо передається конфіденційна інформація, є ризик витоку даних. Потрібно мати на увазі, що дані можуть бути доступні власникам або операторам

сервісу. Також, якщо інтернет підключення погане, або ж серіс недоступний або тимчасово не працює, потрібно буде шукати аналоги для парсингу.

1.6. Постановка завдання

Під час роботи над дипломним проєктом було запропоновано рішення проблеми читабельності JSON формату, який зазвичай зберігається у вигляді рядка. Це допоможе розробникам, які часто працюють з цим форматом збільшити ефективність та швидкість виконання поставлених цілей. Основні завдання, що необхідно виконати:

- Обрати оптимальні технології, середовища розробки та бібліотеки для реалізації десктопного застосунку.

- Розробити інтерфейс та архітектуру застосунку, враховуючи вимоги до продуктивності, функціональності, швидкодії, безпеки інформації та зручності в користуванні.

- Розробити програмну реалізацію десктопного застосунку за допомогою обраних технологій та мов програмування Python, CSS.

- Провести тестування на різних платформах, таких як Windows 10, Windows 11, Linux, MacOS.

- Провести аналіз отриманого проекту та порівняти з уже інсуючими аналогами.

ВИСНОВКИ ДО РОЗДІЛУ 1

JSON (JavaScript Object Notation) та його роль у сучасному програмуванні та обміні даними майже незамінна. JSON є легким, текстовим форматом, який використовується для представлення та передачі структурованих даних. Він є універсальним і широко підтримується багатьма мовами програмування, що робить його популярним вибором для обміну даними між різними системами.

Один з ключових аспектів JSON - це його простота та читабельність для людей. Він використовує зрозумілу для людини структуру, що дозволяє розробникам легко читати, редагувати та розуміти дані. JSON також має широкий спектр підтримуваних типів даних, включаючи числа, рядки, масиви, об'єкти, булеві значення та null.

JSON застосовується в різних сферах, включаючи веб-розробку, мобільні додатки, сервіси API, обмін даними між різними системами та базами даних. Його універсальність і простота роблять його популярним вибором серед більшості розробників. JSON легко інтегрується з багатьма мовами програмування та технологіями. Це дозволяє розробникам ефективно обмінюватися даними між різними системами та компонентами програмного забезпечення.

JSON parser - це інструмент, який дозволяє розбирати (парсити) дані JSON з текстового формату у внутрішню структуру даних, що може бути подальше оброблено в програмі. Парсери JSON дозволяють програмістам легко отримувати доступ до конкретних даних у форматі JSON та виконувати різноманітні операції, такі як валідація, фільтрація, збереження або відображення даних.

РОЗДІЛ 2
ВИБІР ТА ОБГРУНТУВАННЯ ІНСТРУМЕНТІВ ПРОЕКТУВАННЯ

2.1. Середовище розробки

Вибір середовища розробки насправді одне із найголовніших завдань, адже правильно підібрана платформа допоможе скоротити час написання та дасть набагато більші можливості та функціонал. У зв'язку з цим, було розглянуто три основних варіанти: Python IDLE, PyCharm та Visual Studio Code. Перед тим, як прийняти остаточне рішення, хотілось би провести порівняльний аналіз цих середовищ на основі їх переваг та недоліків.

1. Python IDLE

Python IDLE (Integrated Development and Learning Environment) є вбудованим середовищем розробки (IDE), яке постачається разом з офіційною установкою Python. Воно надає базовий набір інструментів для написання та виконання коду Python. По свому вигляду дуже нагадує Power Shell.

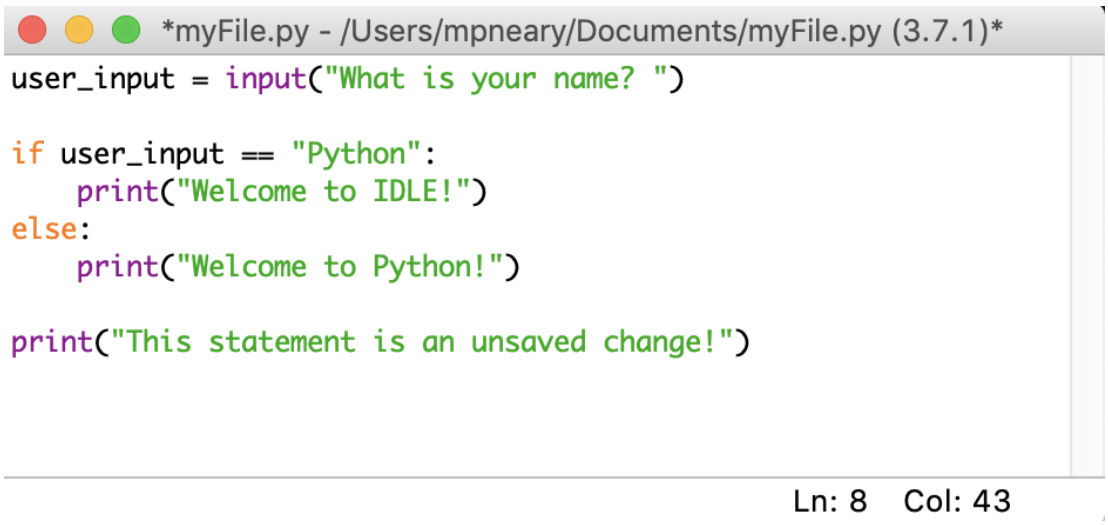

Рис. 2.1. Середовище Python IDLE

Переваги та недоліки Python IDLE:

Переваги:

1. Легкий доступ та простота використання: Python IDLE встановлюється разом з Python і готовий до використання без необхідності встановлювати додаткові компоненти. Він має простий та зрозумілий інтерфейс, що робить його досить легким у навчанні та початку роботи з Python.
2. Редактор коду з підсвіткою синтаксису: Python IDLE має вбудований текстовий редактор, який надає підсвічування синтаксису Python. Це допомагає виділити ключові слова, змінні та інші елементи коду, полегшуючи розуміння та редагування коду.
3. Інтерактивна оболонка: Python IDLE має інтерактивну оболонку, яка дозволяє виконувати код Python по одній команді або блоках. Це дозволяє швидко перевірити результати та експериментувати з кодом без необхідності створювати окремі файлові скрипти.
4. Підтримка оточення виконання коду: Python IDLE підтримує виконання коду в різних режимах, включаючи інтерактивний режим, режим скрипту та режим відладки. Це дозволяє вам виконувати код на виконання крок за кроком, відслідковувати помилки та проводити налагодження.

Недоліки:

1. Обмежена функціональність: Python IDLE має базовий набір функціональності, і його можливості можуть бути обмеженими для складних та великих проектів. Він не має деяких розширених функцій, які присутні в інших повноцінних IDE, таких як автодоповнення коду, система контролю версій, інтегрований термінал та інші.
2. Відсутність розширень та плагінів: Python IDLE не підтримує широкий спектр розширень та плагінів, які можуть розширити його

функціональність або налаштувати його під конкретні потреби розробки. Це може бути обмеженням для більш вимогливих проектів або специфічних сценаріїв.

3. Обмежена підтримка проектів: Python IDLE не надає широкої підтримки для організації та керування проектами. Він не має інтегрованої системи керування проектами, розподілу коду на модулі та пакети, що може вплинути на організацію та розвиток складних проектів.
4. Вимоги до продуктивності: Python IDLE може бути менш продуктивним для великих проектів або при використанні великих обсягів даних, оскільки його продуктивність може бути обмеженою. Він може відчувати затримки при обробці великих файлів або складних операцій.

Python IDLE є хорошим початковим варіантом для новачків у Python або для простих проектів. Однак, для складніших та більших проектів рекомендується розглянути використання інших потужних IDE, таких як PyCharm, Visual Studio Code або Spyder, які надають більшу функціональність та розширені можливості розробки.

2. PyCharm

PyCharm - це одна з найпопулярніших професійних інтегрованих середовищ розробки (IDE) для Python, розроблена компанією JetBrains. Вона надає розширений набір інструментів та функціональності, спеціально створених для ефективної розробки на Python.

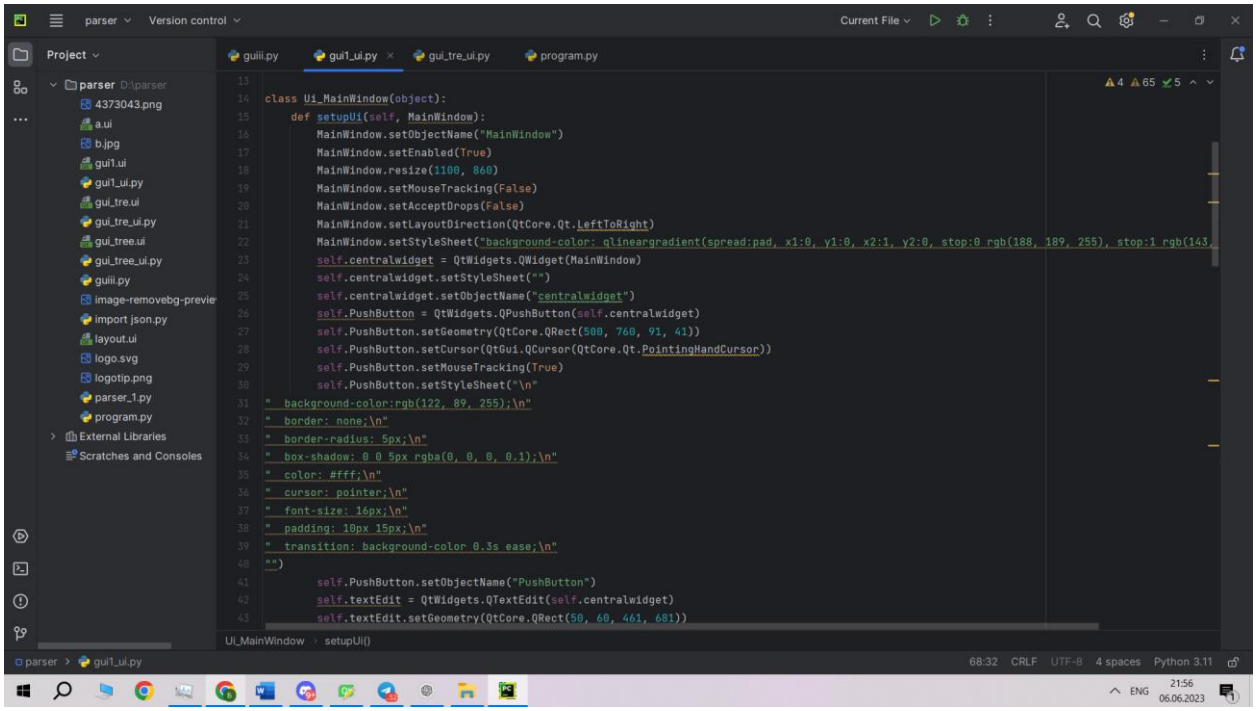

Рис. 2.2. Середовище PyCharm

Переваги та недоліки PyCharm:

Переваги:

1. Розширений набір функціональності: PyCharm має велику кількість функцій та інструментів, які полегшують розробку на Python. Це включає автодоповнення коду, перевірку синтаксису, рефакторинг, систему контролю версій, інтегрований термінал та багато іншого. Вона допомагає покращити продуктивність та ефективність розробника.
2. Система автодоповнення та інтелектуальний аналіз коду: PyCharm має потужну систему автодоповнення, яка пропонує варіанти завершення коду на основі контексту. Вона також включає інтелектуальний аналіз коду, який виявляє помилки, непотрібний код та пропонує покращення.
3. Налагоджувальник: PyCharm надає вбудований налагоджувальник, який дозволяє крок за кроком виконувати та налагоджувати код. Він має вбудовані панелі для перегляду значень змінних, виклику функцій, встановлення точок зупину та багато іншого, що спрощує відлагодження програм.

4. Підтримка віртуальних середовищ: PyCharm дозволяє легко створювати та керувати віртуальними середовищами Python, що дозволяє ізолювати залежності та проекти. Це допомагає уникнути конфліктів та забезпечити чистоту робочого середовища.

Недоліки:

1. Великі вимоги до ресурсів: PyCharm відносно важке середовище, яке може вимагати значних обсягів оперативної пам'яті та потужності процесора. На старих або менш потужних комп'ютерах це може впливати на продуктивність роботи.
2. Великий обсяг функціональності для новачків: PyCharm має багато функцій та налаштувань, що можуть бути перевагою для досвідчених розробників, але можуть здаватися складними для новачків, які тільки починають вивчати Python.
3. Вартість платної версії: Є дві версії PyCharm - Community Edition (безкоштовна) та Professional Edition (платна). Деякі розширені функції доступні тільки в платній версії, що може бути обмеженням для деяких користувачів з обмеженим бюджетом.

PyCharm - це потужне та розширене IDE для розробки на Python, ідеально підходить для серйозних проектів та професійного використання. Вона надає широкий спектр функцій, що полегшують розробку та покращують продуктивність розробника [9].

3. Visual Studio Code

Visual Studio Code (VS Code) - це безкоштовне інтегроване середовище розробки (IDE), розроблене компанією Microsoft. Це популярний інструмент для розробки програмного забезпечення, включаючи розробку на мові Python. Загалом, інтерфейс дуже дружній навіть до людей які не займались програмуванням.

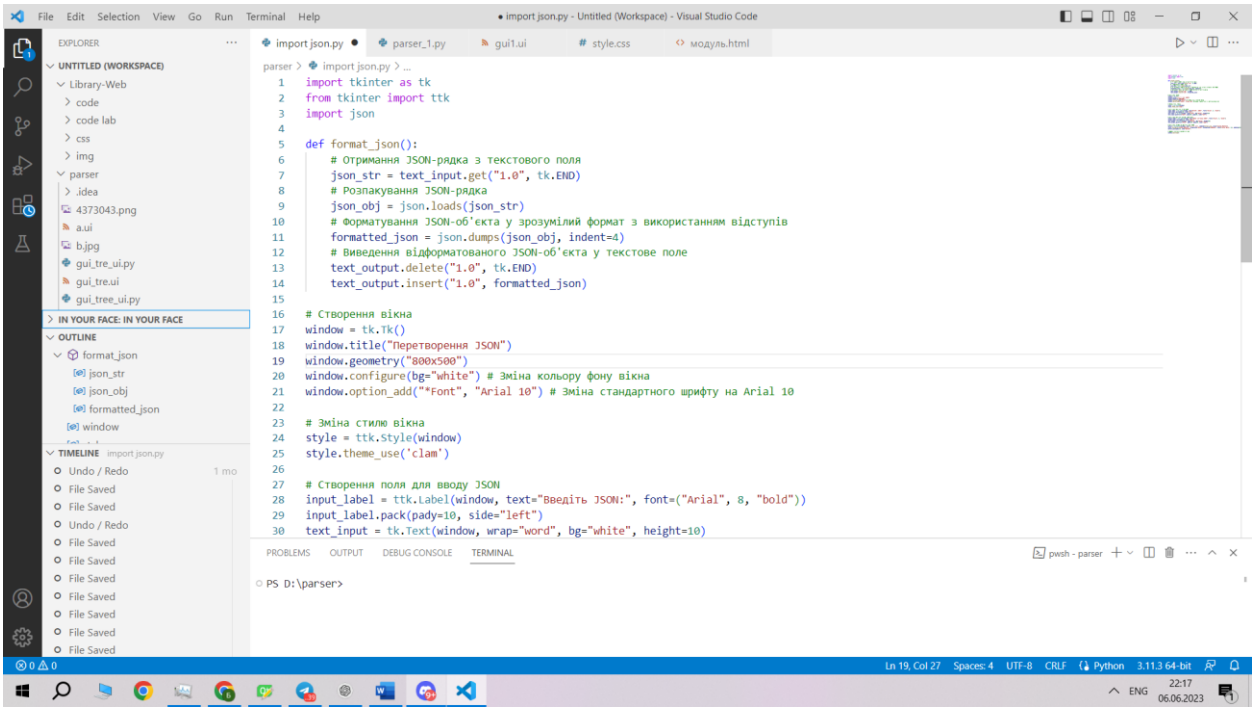

Рис. 2.3. Інтерфейс Visual Studio Code

Переваги та недоліки Visual Studio Code:

Переваги:

1. Розширюваність: VS Code має велику кількість розширень, що дозволяють розширити його функціональність та налаштувати його під ваші потреби. Це включає плагіни для автодоповнення, синтаксичного підсвічування, систем контролю версій, налагоджування та багато іншого. Розширення розробляються спільнотою, що дозволяє вам знайти інструменти, які відповідають вашим потребам.
2. Легкість використання та налаштування: VS Code має дружній та зрозумілий інтерфейс, що дозволяє легко навігувати, редагувати та відлагоджувати код. Він також має просту систему налаштувань, що дозволяє змінювати різні параметри та вигляд IDE, адаптуючи його під ваший стиль розробки.
3. Інтегрована система контролю версій: VS Code має вбудовану підтримку систем контролю версій, таких як Git. Є можливість здійснювати коміти, переглядати різницю, розрішувати конфлікти та

багато іншого, прямо з IDE. Це полегшує роботу над проектами та керування версіями коду.

4. Відлагодження та налагоджувальник: VS Code надає вбудований налагоджувальник з можливістю крок за кроком виконувати код та відстежувати значення змінних. Він також має інструменти для пошуку помилок та відлагодження коду з використанням точок зупину.

Недоліки:

1. Вимоги до ресурсів: В порівнянні з легшими текстовими редакторами, VS Code може вимагати більше ресурсів системи, особливо при роботі з великими проектами або використанні багатьох розширень.
2. Потребує налаштування: Всупереч його легкому використанню, деякі функції VS Code можуть потребувати налаштування та знань для їх належного використання. Налаштування вимагає деякої кількості часу та зусиль, щоб настроїти IDE під ваши потреби.
3. Обмежена функціональність: У порівнянні з деякими повноцінними IDE, VS Code може мати обмежену функціональність. Наприклад, він може не мати певних інструментів або функцій, які присутні в більш спеціалізованих IDE.
4. Підтримка мови Python: В хоча і VS Code є популярним середовищем розробки для різних мов програмування, включаючи Python, він може не мати такої повної підтримки Python, як деякі спеціалізовані IDE, такі як PyCharm.

Загалом, Visual Studio Code є потужним та гнучким інструментом для розробки на мові Python. Він надає багато функцій та можливостей, які полегшують розробку та підвищують продуктивність розробників. Проте, залежно від потреб та вподобань, ви також можна розглянути інші варіанти IDE для Python для знаходження оптимального середовища розробки.

2.2. Вибір бібліотек

Вибір бібліотеки має велике значення у написанні проекту. Перш за все, бібліотеки надають готові рішення для різних завдань, що значно прискорює процес розробки. Замість того, щоб писати код з нуля, можна скористатися функціональністю, яку надає бібліотека. Це дозволяє економити час та зусилля розробника.

Крім того, вибір правильної бібліотеки може вплинути на продуктивність, якість та розширюваність проекту. Якщо обрана бібліотека добре оптимізована та має хорошу підтримку, це може позитивно вплинути на швидкодію та стабільність програми. Крім того, якщо бібліотека активно розвивається та має велике співтовариство користувачів, це забезпечує наявність документації, підтримки та відповідей на питання, що спрощує роботу розробників.

Також, правильний вибір може вплинути на архітектуру проекту. Деякі бібліотеки пропонують різні підходи та структури для розв'язання проблем, і вибір відповідної бібліотеки може визначити загальну структуру та організацію проекту.

Різні бібліотеки можуть мати свої особливості, функціональність та можливості. Важливо вибрати таку бібліотеку, яка найкраще задовольняє потреби проекту та дозволяє досягти поставлених цілей.

Існує кілька способів встановлення бібліотек на Python:

1. Використання пакетного менеджера pip: pip є стандартним пакетним менеджером для Python. Він дозволяє легко встановлювати пакети з репозиторію PyPI (Python Package Index). Для встановлення бібліотеки використовується команда pip install у командному рядку, наприклад: pip install «назва_бібліотеки».
2. Використання pipenv або poetry: Ці інструменти дозволяють створювати ізольовані середовища для проекту та керувати залежностями. Вони автоматично встановлюють необхідні пакети

згідно з файлом залежностей (наприклад, Pipfile для pipenv або pyproject.toml для poetry).

3. Використання системи керування пакетами ОС: Деякі пакети можуть мати пакети для встановлення через систему керування пакетами вашої операційної системи, такі як apt, yum, brew тощо. Це може бути корисно, якщо бібліотека має залежності, які також потрібно встановити.
4. Використання вбудованих засобів розробки середовища: Деякі IDE мають вбудовані менеджери пакетів, які дозволяють встановлювати бібліотеки безпосередньо з інтерфейсу користувача.

Для виконання поставленої задачі були обрані бібліотеки PyQt5, Qt, json, pyjq, psycopg2. Вони можуть дуже допомогти в реалізації проекту.

1. Json

У Python існує вбудована бібліотека json, яка надає можливості для роботи з даними у форматі JSON. Ця бібліотека дозволяє здійснювати серіалізацію (перетворення об'єктів Python у рядок JSON) та десеріалізацію (перетворення рядка JSON у об'єкти Python).

Основні функції, які надає бібліотека json в Python, включають:

1. json.dumps(): Ця функція дозволяє серіалізувати об'єкти Python у рядок JSON. Вона приймає об'єкт Python та повертає рядок JSON.
2. json.loads(): Ця функція дозволяє десеріалізувати рядок JSON у об'єкти Python. Вона приймає рядок JSON та повертає відповідний об'єкт Python.
3. json.dump(): Ця функція дозволяє серіалізувати об'єкти Python у файл у форматі JSON. Вона приймає об'єкт Python та файловий об'єкт і записує у файл JSON-подібний вміст.
4. json.load(): Ця функція дозволяє десеріалізувати JSON-подібний вміст з файлу у відповідні об'єкти Python. Вона приймає файловий об'єкт та повертає об'єкт Python, який містить дані з файлу.

Бібліотека json також надає підтримку для специфічних опцій та параметрів, таких як сортування ключів, обробка спеціальних значень (наприклад, NaN, Infinity) та контроль форматування виведеного JSON. Вона дозволяє легко працювати зі структурованими даними у форматі JSON в Python та забезпечує зручність та ефективність взаємодії з цим форматом даних.

2. pyjq

Бібліотека pyjq є обгорткою Python для JQ, потужного інструменту обробки та маніпулювання даними у форматі JSON. JQ є мовою запитів та фільтрації, спеціально розробленою для JSON, і pyjq дозволяє використовувати цю мову в середовищі Python.

Основна функція pyjq полягає у забезпеченні простого та зручного способу виконання запитів до структурованих даних у форматі JSON. Вона дозволяє вибирати, фільтрувати та перетворювати дані, що допомагає ефективно обробляти складні структури даних.

Однією з можливостей pyjq є фільтрація та вибірка даних. Тобто, можна використовувати запити, умови та операції для відбору певних елементів зі структури JSON. Це дозволяє вам швидко отримувати потрібні дані та здійснювати подальшу обробку.

Крім того, pyjq дозволяє змінювати та перетворювати структуру даних JSON. Можна додавати, видаляти та змінювати поля та значення, а також здійснювати групування даних та інші маніпуляції. Це допомагає вам адаптувати дані вашого проекту та здійснювати необхідні перетворення.

3. Qt

Qt - це потужна бібліотека для розробки крос-платформенних програм. Вона пропонує широкий набір інструментів і можливостей для створення графічних інтерфейсів користувача (GUI), мережевих додатків, базової

функціональності програм і багато іншого. Основні риси Qt включають наступне:

- Крос-платформенність: Qt дозволяє розробляти програми, які працюють на різних операційних системах, таких як Windows, macOS, Linux, Android і iOS. Це забезпечує широку аудиторію користувачів і зменшує зусилля, необхідні для портативності програм.
- Графічний інтерфейс користувача (GUI): Qt має потужні засоби для розробки інтуїтивно зрозумілих і привабливих графічних інтерфейсів. Він надає різноманітні елементи керування (кнопки, поля вводу, таблиці тощо), можливості мультимедіа, анімації, стилізації і кастомізації.
- Мережева підтримка: Qt пропонує потужні функції для розробки мережевих додатків. Можна створювати клієнт-серверні програми, робити запити до веб-серверів, використовувати протоколи комунікації (HTTP, TCP/IP, UDP) і реалізовувати мережеві функції в своїх програмах.
- Базова функціональність: Qt надає широкий набір базових класів і функцій, які допомагають вам в роботі зі звичайними задачами, такими як робота з файлами і каталогами, робота з базами даних, обробка подій, потоки і багатопоточність, серіалізація даних і багато іншого.
- Інструменти розробки: Qt має велику набір інструментів для підтримки розробки, включаючи інтегроване середовище розробки (IDE) Qt Creator, засоби візуального проектування і редактори інтерфейсів користувача. Ці інструменти сприяють швидкому створенню програм і полегшують процес розробки.

Qt активно використовується в індустрії для створення різних типів додатків, включаючи десктопні програми, мобільні додатки, ігри, інтерфейси вбудованих систем, програми для Інтернету речей та багато іншого. Цей

фреймворк має велику спільноту розробників, що забезпечує підтримку, документацію, приклади та інші ресурси для допомоги у розробці програм з використанням Qt.

4. PyQt5

PyQt5 - це потужна бібліотека для створення графічного інтерфейсу користувача (GUI) в мові програмування Python. Вона базується на бібліотеці Qt, яка надає розширені можливості для розробки крос-платформенних додатків з використанням різних платформ, таких як Windows, macOS, Linux та інші.

PyQt5 надає доступ до широкого набору компонентів та функцій Qt. Можна створювати вікна, діалогові вікна, меню, кнопки, тексти, таблиці, списки, вкладки, значки та багато іншого. Крім того, можна використовувати властивості та методи цих компонентів для налаштування їх поведінки.

Бібліотека базується на механізмі подій та сигналів, що дозволяє відреагувати на дії користувача або зміни стану програми. Є можливість призначати обробники подій, які будуть викликатися при виникненні певних подій, таких як натискання кнопки чи зміна значення поля введення.

При використанні PyQt5 рекомендує використовувати паттерн MVC для організації коду. Це дозволяє виділити логіку програми в окремі моделі, керувати відображенням інтерфейсу користувача та забезпечувати взаємодію між ними. Основна задумка паттерна MVC полягає в розподілі відповідальностей між компонентами:

- **Модель** представляє бізнес-логіку та управляє даними. Вона відповідає за обробку та збереження даних, виконання операцій над ними, включаючи читання, запис, оновлення та видалення. Модель не залежить від інших компонентів паттерна.
- **Представлення** відображає дані користувачу та взаємодіє з ним. Воно отримує дані з моделі та відображає їх у відповідному форматі. Представлення може також відстежувати події та відправляти їх до контролера для обробки.

- **Контролер** взаємодіє як з моделлю, так і з представленням. Він приймає вхідні дані від користувача через представлення, виконує відповідні дії та оновлює модель. Контролер також може оновлювати представлення залежно від змін у моделі. Він служить посередником між моделлю та представленням.

MVC чітко відокремлює логіку програми, користувацький інтерфейс та управління даними. Це дозволяє зберігати код більш організованим, масштабованим та легким для змін.

Крім того, використання паттерна MVC сприяє повторному використанню коду, оскільки модель може бути використана в різних представленнях або контролерах. Крім того, зміни в логіці програми або в інтерфейсі користувача можуть відбуватися незалежно один від одного, що полегшує розвиток та тестування програмного забезпечення.

Також бібліотека PyQt5 дає змогу розробляти крос-платформенні програми. Тобто, написавши код один раз - можна запускати його на різних операційних системах без змін, що безсумнівно є дуже великим плюсом у сучасній розробці.

PyQt5 постачається з візуальним дизайнером Qt Designer.

Qt Designer - це потужний графічний інструмент для візуального проектування та налаштування інтерфейсу користувача для програм на основі Qt. Він надає зручний спосіб створення та налаштування вікон, віджетів та їх властивостей без необхідності писати код вручну.

Основна перевага Qt Designer полягає в його інтуїтивному візуальному інтерфейсі. Можна просто перетягувати віджети, такі як кнопки, поля вводу, таблиці, і розміщувати їх на формі. За допомогою різних інструментів та панелей властивостей, можна встановлювати розмір, положення, шрифт, кольори та інші атрибути віджетів.

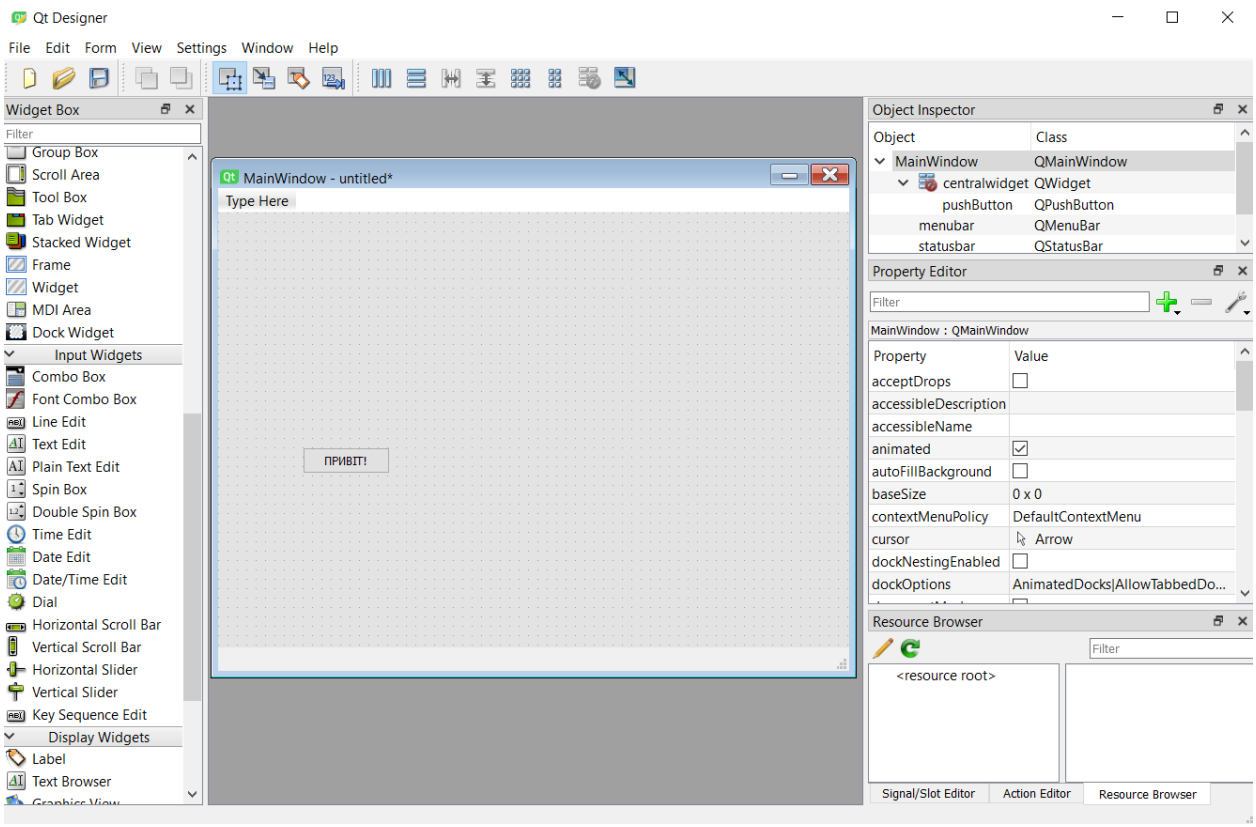

Рис. 2.4. Інтерфейс Qt Designer

Qt Designer також підтримує повний набір віджетів, доступних в Qt. Можна вибрати віджет, який потрібен, та налаштувати його властивості безпосередньо у редакторі. Крім того, є можливість додавати сигнали та слоти до віджетів, що дозволяє підключати їх до відповідних функцій коду для обробки подій.

Одна з переваг Qt Designer полягає у можливості підключати інтерфейс до коду. Тобто, є можливість згенерувати заготовки коду на основі дизайну та підключити їх до програми, написаної на мовах, таких як C++ або Python. Це спрощує інтеграцію між візуальною та логічною частинами програми.

Qt Designer також може налаштовувати стилі та теми інтерфейсу. Розширення надає можливість вибрати з великого набору готових стилів або налаштувати їх самостійно, щоб забезпечити бажаний вигляд та взаємодію з програмою.

Після того, як розробку інтерфейсу за допомогою Qt Designer закінчено, присутня можливість експортувати дизайн у файл формату .ui,

який можна згодом використати в проекті або ж конвертувати за допомогою наприклад команди pyuic в готовий код інтерфейсу на Python. Це дає можливість поділитися дизайном з іншими розробниками або використовувати його в різних проектах. [10]

5. Psycopg2

Psycopg2 - це потужна бібліотека для Python, яка дозволяє розробникам зручно працювати з базами даних PostgreSQL. Основні особливості цієї бібліотеки включають:

- Psycopg2 легко інтегрується з проектами Python і дозволяє розробникам зручно використовувати її функціонал у своїх програмах.
- Бібліотека повністю використовує можливості PostgreSQL і надає доступ до всіх його функцій, таких як операції з JSON-даними та розширеннями.
- Psycopg2 оптимізована для швидкої роботи з PostgreSQL, що дозволяє ефективно виконувати запити до бази даних та отримувати результати.
- Бібліотека надає можливість працювати з транзакціями в PostgreSQL, що дозволяє забезпечити цілісність та безпеку даних.
- Psycopg2 має вбудовані механізми для захисту від атак SQL-ін'єкцій, що дозволяє безпечно виконувати запити до бази даних.
- Бібліотека підтримує використання курсорів, що дозволяє ефективно обробляти великі набори даних і виконувати складні запити.

ВИСНОВКИ ДО РОЗДІЛУ 2

Після ретельного порівняльного аналізу трьох середовищ розробки для Python - Python IDLE, PyCharm і Visual Studio Code, я прийшов до висновку, що PyCharm є найкращим варіантом в моєму випадку.

PyCharm пропонує широкий набір функціональності, яка полегшує розробку та покращує продуктивність розробника. Він має потужні інструменти автодоповнення коду, систем контролю версій, налагоджувача та вбудованої підтримки віртуальних середовищ. Ці функції допоможуть в розробці складного проекту та підтримці кодової бази.

PyCharm надає зручний та дружній інтерфейс користувача, що дозволить ефективно працювати з кодом, навігувати по проекту та виконувати налагодження. Його вбудована система контролю версій, зокрема з підтримкою Git, дозволить легко керувати версіями коду.

Крім того, PyCharm - це платформа, спеціалізована для розробки на мові Python. Це означає, що воно надає широкий набір інструментів та підтримку, специфічну для Python, що спростить роботу з цією мовою програмування.

В загальному, PyCharm відповідає усім необхідним потребам. Його розширена функціональність, спеціалізація на мові Python, зручний інтерфейс та підтримка системи контролю версій роблять його найкращим варіантом для виконання проекту.

Для виконання поставленої цілі були обрані бібліотеки PyQt5, Qt, json, pyjq, psycopg2. Вони можуть дуже допомогти в реалізації проекту.

РОЗДІЛ 3
СТВОРЕННЯ ІНТЕРФЕЙСУ ТА АЛГОРИТМУ ПРОГРАМИ

3.1. Проектування в Qt Designer

Qt Designer - це потужний графічний інструмент для візуального проектування та налаштування інтерфейсу користувача для програм на основі Qt. Він надає зручний спосіб створення та налаштування вікон, віджетів та їх властивостей без необхідності писати код вручну.

Щоб почати роботу необхідно створити новий файл, обрати Main Window та натиснути на Create.

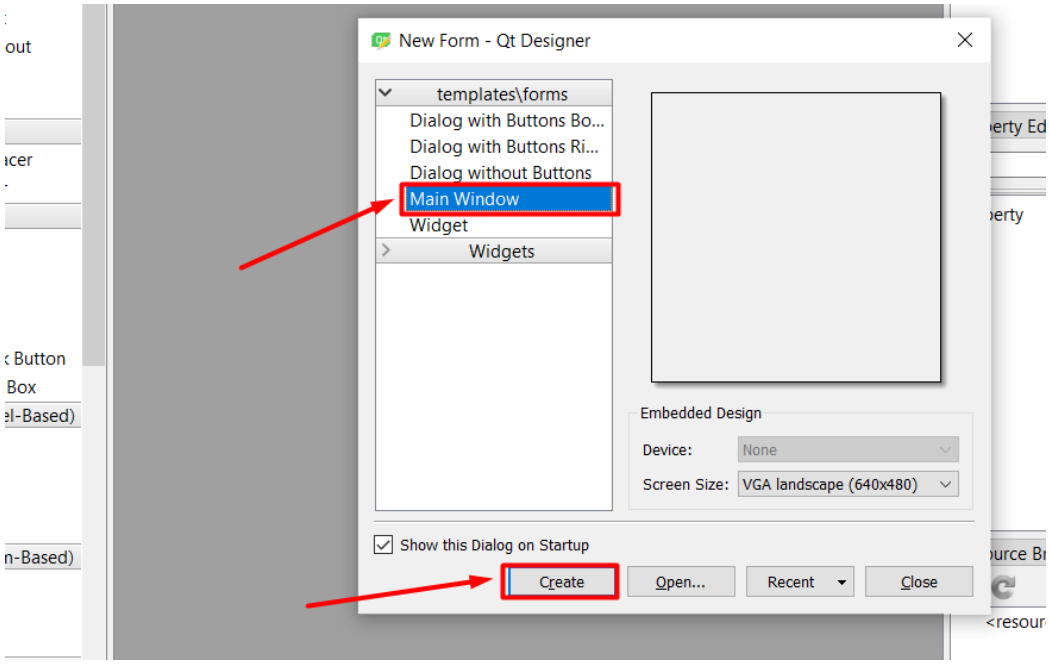

Рис. 3.1. Створення нового пустого вікна

Після цього відкривається можливість пересовувати віджети, міняти їх властивості, форми та стилі. Для виконання поставленого проекту перш за все потрібне поле для редагування тексту. Це можна зробити за допомогою віджета textEdit.

"TextEdit" в Qt Designer - це елемент керування (widget), який використовується для відображення та редагування текстового вмісту. Він надає можливість користувачам вводити та змінювати текст.

TextEdit може бути використаний для редагування однорядкового або багаторядкового тексту, залежно від налаштувань елемента керування. Він підтримує такі функції, як вставка, видалення, копіювання та переміщення тексту, а також може виконувати операції з форматуванням тексту, такі як налаштування шрифту, колірів, вирівнювання тексту тощо.

Зважаючи на ідею проекту, може знадобитись два таких віджета, в одному корирстувач буде вводити текст, а в іншому алгоритм буде видавати готовий після парсингу JSON рядок.

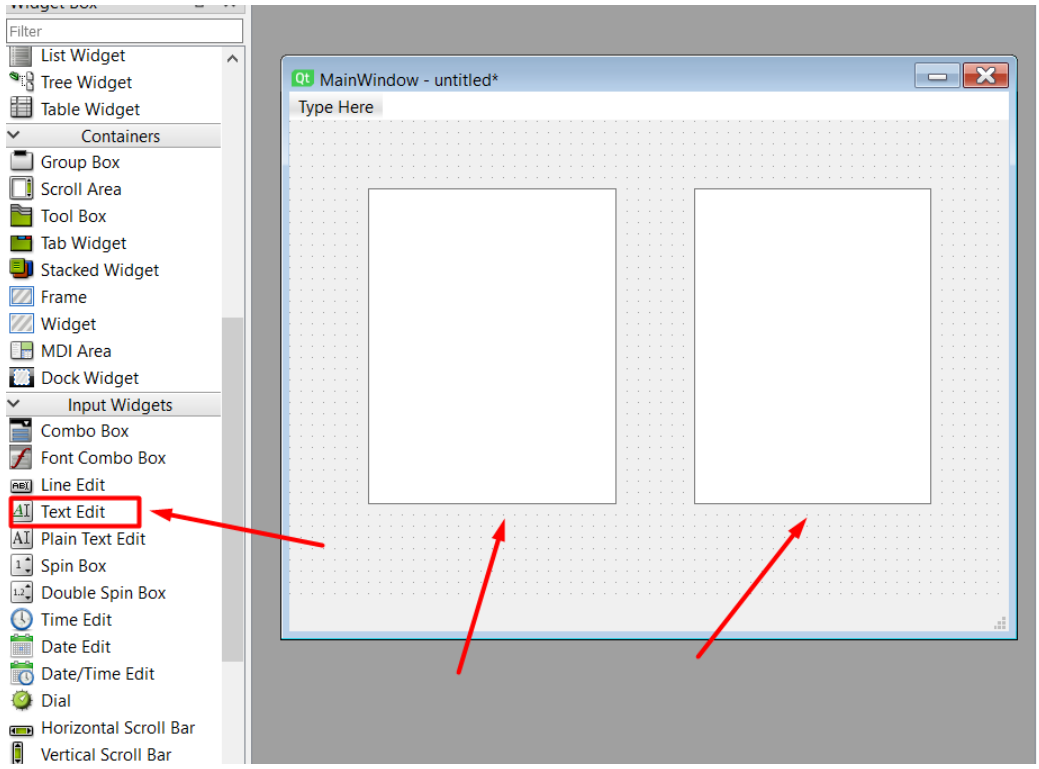

Рис. 3.2. Приблизне розташування текстових полів

Далі потрібно створити кнопки які будуть відповідати за певні дії, наприклад Copy та Clear які зроблять використання додатку для користувача легше і функціональніше, що позитивно повпливає на кінцевий результат програми.

Зробити це можна за допомогою віджета Push Button "PushButton" в Qt Designer - це елемент керування (widget), який використовується для створення кнопок у графічному інтерфейсі користувача. PushButton надає можливість взаємодіяти з програмою, натисканням на кнопку.

Кнопка PushButton може мати текст або іконку, які відображаються на кнопці. Також можна налаштовувати вигляд кнопки, включаючи розміри, колір, стиль тексту та фоновий колір кнопки.

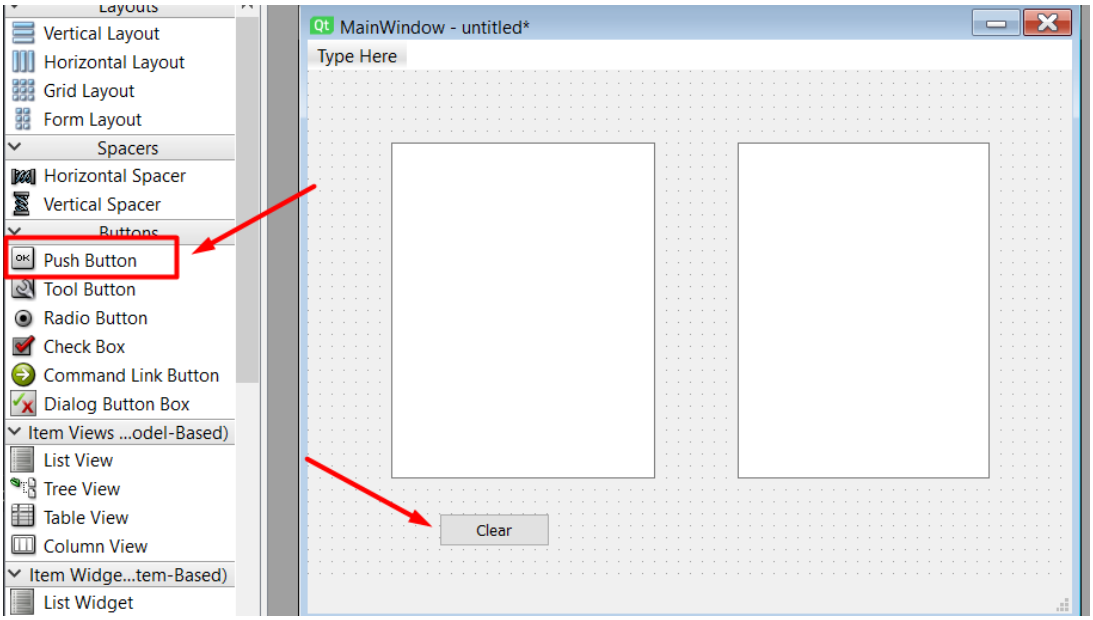

Рис. 3.3. Розташування кнопки Clear

Використовуючи властивості, доступні в Qt Designer можна визначити сигнали та слоти для взаємодії з кнопкою у коді, коли форма виконується. Для цього потрібно обрати Edit Signals/Slots та провести стрілку від кнопки до віджета. Так як кнопка Clear повинна очищувати відразу два поля, відповідно потрібно провести дві стрілки.

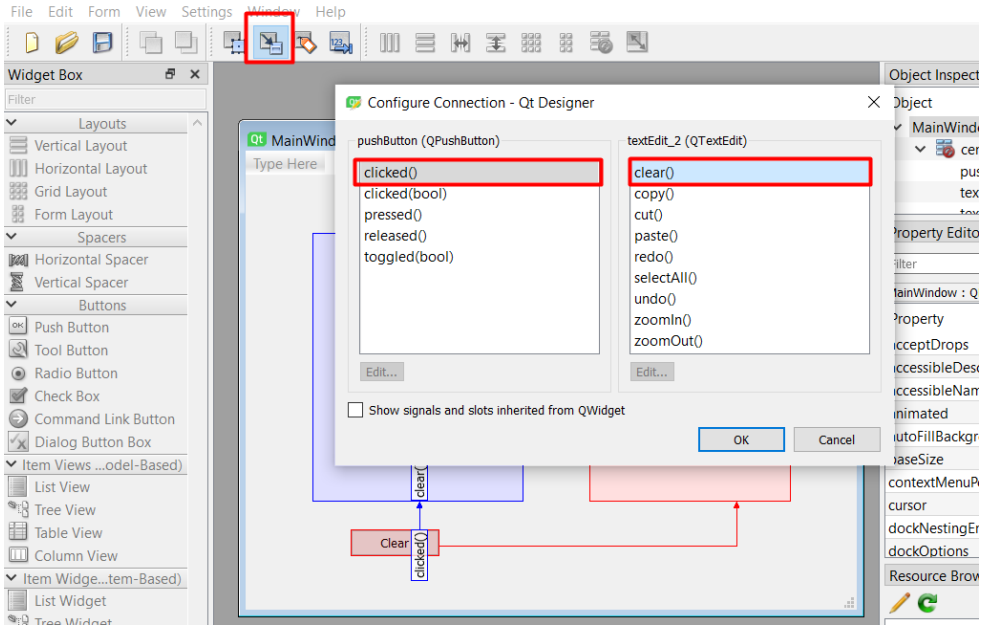

Рис. 3.4. Настройка функціоналу кнопки Clear

Так як потрібно, щоб поля очищувались при натисканні на кнопку, вибираємо функцію onclicked() та обираємо clear()

Додамо заготовки без функціоналу для кнопок Parse і View, вони знадобляться пізніше при написанні алгоритмів програми.

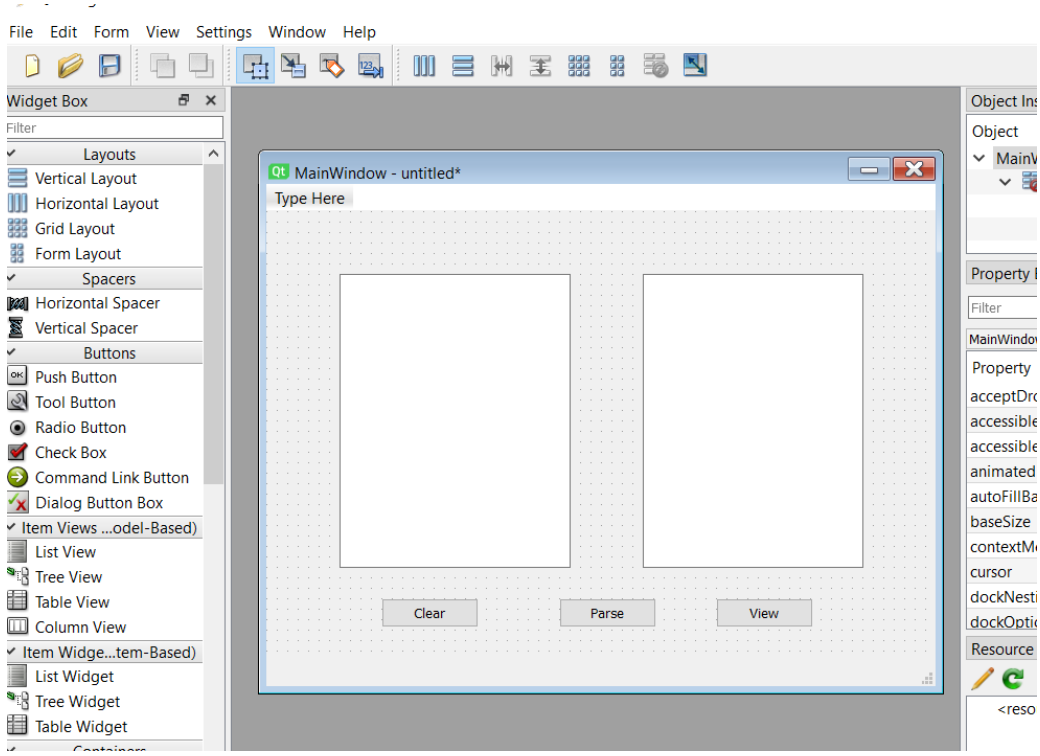

Рис. 3.5. Розташування всіх необхідних кнопок

Для покращення функціоналу необхідного для роботи парсера потрібно додати treeWidget.

TreeWidget в Qt - це елемент керування, який надає ієрархічний список даних у вигляді дерева. Він дозволяє відображати дані у вигляді вузлів і підвузлів, що дозволяє користувачам інтерактивно розгортати та згортати елементи, вибирати певні елементи або виконувати дії з ними.

TreeWidget має такі основні можливості:

- Елементи TreeWidget організовані у вигляді дерева, де кожен вузол може мати декілька дочірніх елементів. Це дозволяє представляти складні структури даних і групувати їх за рівнями або категоріями.
- Користувач може редагувати текстові поля елементів TreeWidget безпосередньо, а також вибирати один або декілька елементів за допомогою миші або клавіатури.
- Елементи TreeWidget можуть відображати як текст, так і іконки або комбінацію обох. Це дозволяє використовувати іконки для позначення стану або типу елементу.
- Можна реагувати на події, пов'язані з TreeWidget, такі як вибір елемента, подвійне натискання мишею, зміна значень тощо. Тобто можна використовувати сигнали та слоти для обробки цих подій та виконання потрібних дій.
- TreeWidget надає можливість сортувати та фільтрувати елементи за різними критеріями, наприклад налаштовування властивості сортування та фільтрації, щоб відобразити лише певні елементи або впорядкувати їх у певному порядку.

TreeWidget можна використовувати для створення дерева каталогів, дерева об'єктів, структури документа, дерева категорій тощо. В даному випадку цей віджет допоможе відобразити ієрархію JSON рядка: ключі, дані, масиви, інші JSON рядки тощо.

Розташуємо treeWidget відразу за полем результату, щоб зробити можливим перемикання вигляду з простого структурованого тексту в деревовидний формат.

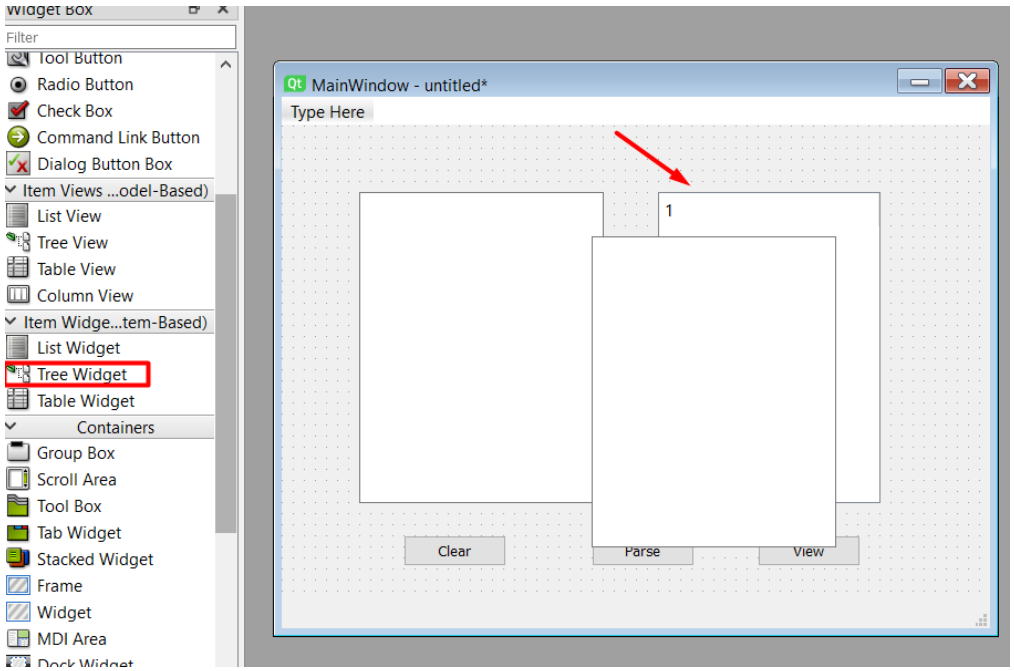

Рис. 3.6. Розміщення treeWidget

Наступним кроком буде додавання layout в інтерфейс парсера.

В Qt Designer, "layout" (розкладка) використовується для організації розташування та вигляду елементів у вікні або формі. Layouts допомагають автоматично розміщувати елементи відповідно до певних правил і забезпечують адаптивність інтерфейсу під різні розміри вікон або зміни розташування елементів.

Qt Designer надає кілька типів розкладок, таких як:

- QVBoxLayout (Vertical Layout): Розміщує елементи вертикально один за одним. Елементи розміщуються один над іншим.
- QHBoxLayout (Horizontal Layout): Розміщує елементи горизонтально один поруч з одним. Елементи розміщуються один за іншим.
- QGridLayout (Grid Layout): Розміщує елементи у вигляді сітки з рядками та стовпцями. Елементи розміщуються у визначених рядках та стовпцях.

- QFormLayout (Form Layout): Розміщує елементи у вигляді форми, з мітками зліва та елементами вводу або іншими елементами справа.
- QVBoxLayout і QHBoxLayout можуть бути також вкладені один в одного, щоб створити більш складні розкладки.

Використання розкладок дозволяє легко керувати розташуванням та поведінкою елементів в інтерфейсі користувача, забезпечуючи гнучкість та легкість відлагодження.

Необхідно створити Horizontal Layout та розташувати елементи кнопки у потрібному порядку.

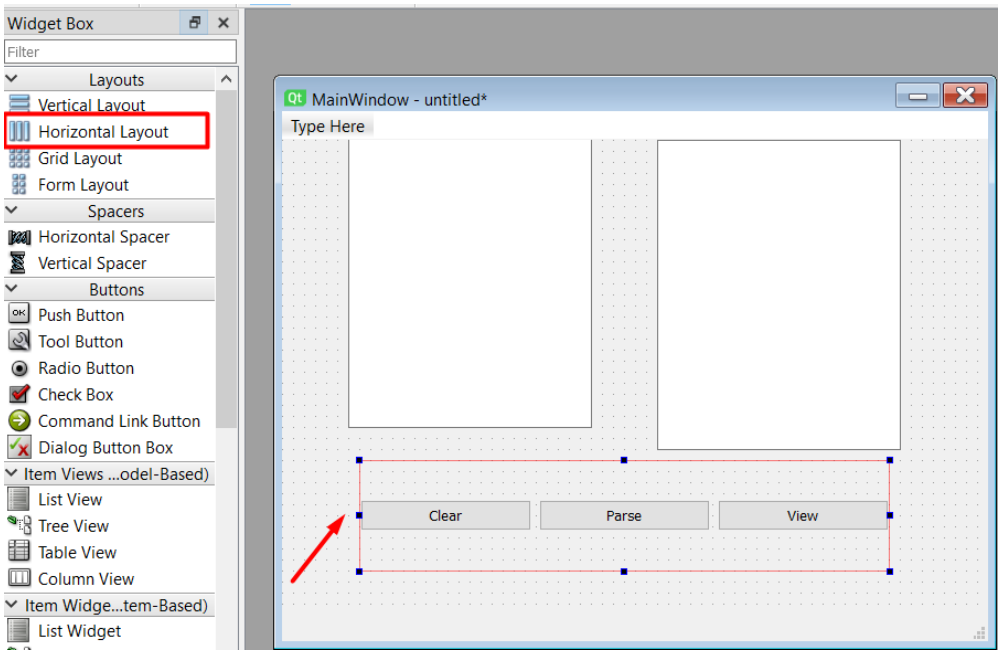

Рис. 3.7. Розташування кнопок на horizontal layout

Для налаштування відступів між елементами розкладки, можна змінити числа у полях Layout Margin та Layout Spacing.

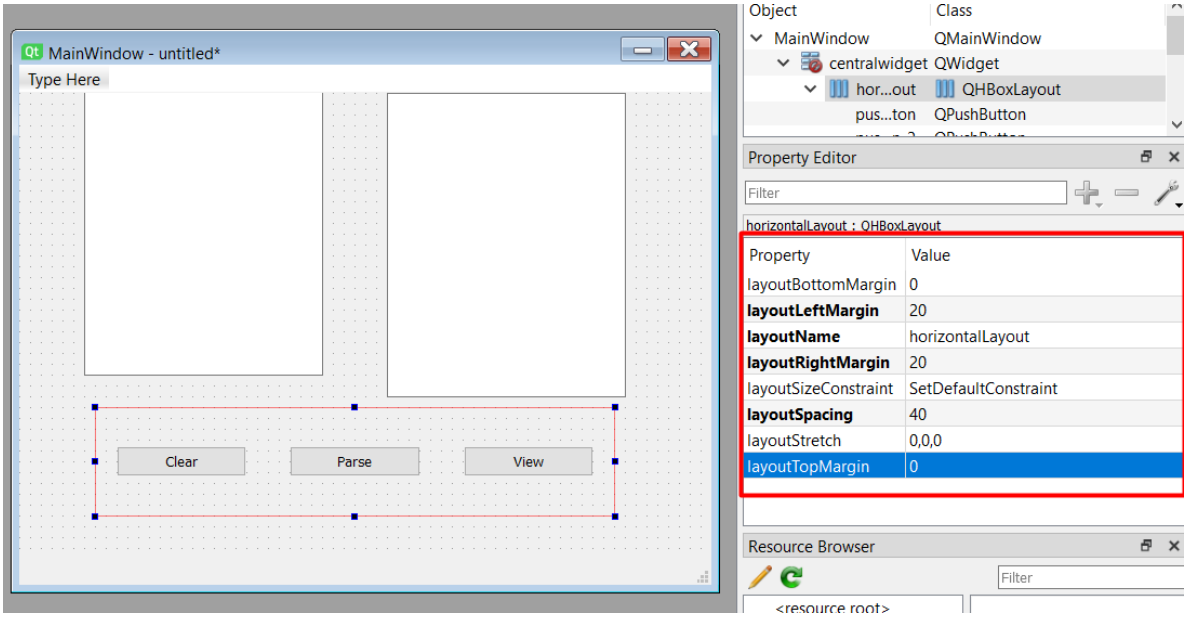

Рис. 3.8. Настройка відступів у layout

При необхідності можна також змінити значення layoutStretch, наприклад зробити головну кнопку Parse більше інших.

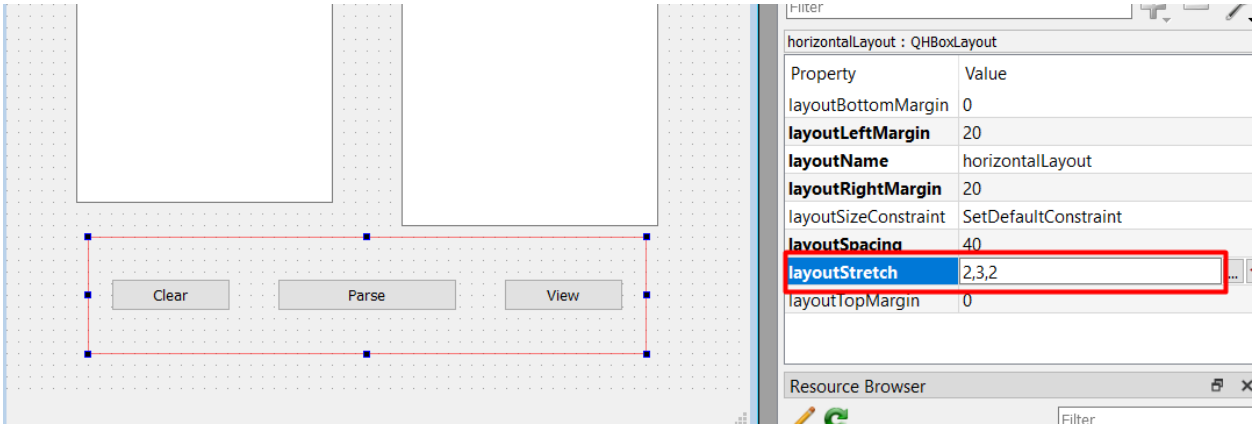

Рис. 3.9. layoutStretch

Для того щоб layouts працювали коректно, необхідно щоб всі елементи були в одному layout. Для цього потрібно створити ще дві розкладки, перша розкладка(horizontal) для налаштування відстані між полями вводу JSON рядка та результату.

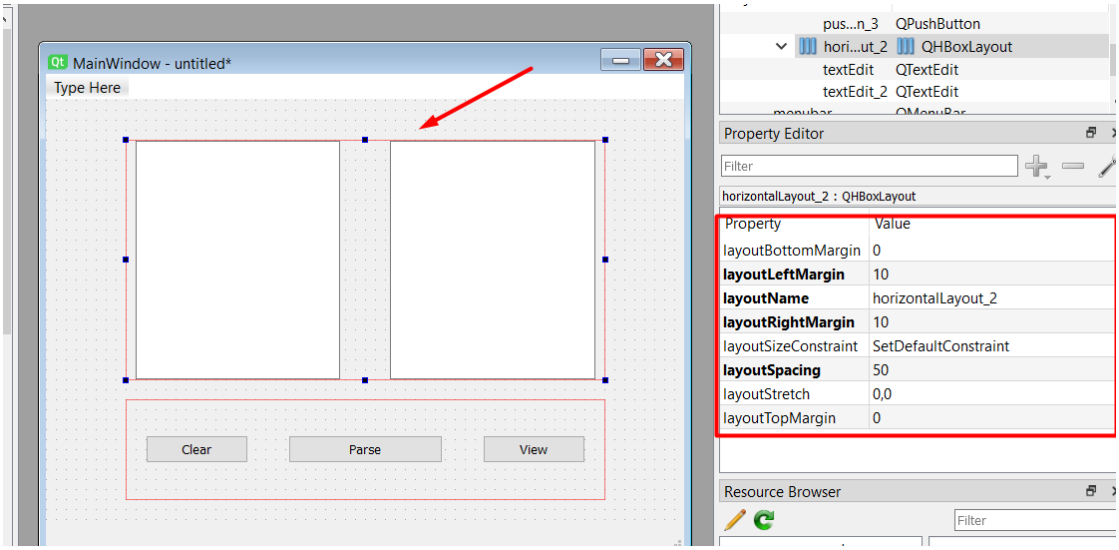

Рис. 3.10. Horizontal Layout для полів вводу та результату

Далі потрібно додати все в Vertical Layout, для того щоб вибрані елементи, а саме розкладка з кнопками стояла нижче розкладки з полями. Також потрібно відредагувати layoutStretch відповідно до завдання.

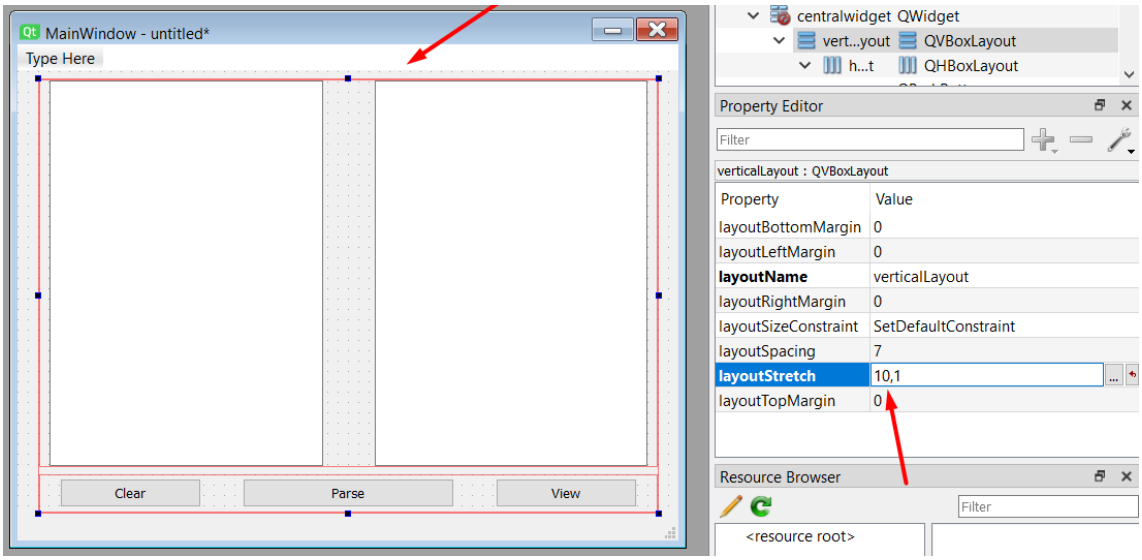

Рис. 3.11. Готова розкладка віджетів

Все що залишилось зробити, це натиснути правою кнопкою у вільномі місці centralWidget(вікна) та вибрати опцію Lay out -> Lay out horizontally та зберегти обраний файл в форматі «.ui».

3.2. Перетворення .ui в python формат

Pyuic5 (Python User Interface Compiler for Qt 5) - це інструмент командного рядка, який поставляється разом з бібліотекою PyQt5. Він використовується для перетворення файлів інтерфейсу користувача, створених у Qt Designer у форматі .ui, в вихідний код Python.

Pyuic5 дозволяє автоматично генерувати Python-код, який відповідає інтерфейсу, створеному у Qt Designer. Це дозволяє розділити розробку графічного інтерфейсу та програмного коду, тому що можна створювати і змінювати інтерфейс у зручному для розробника візуальному редакторі (Qt Designer), а потім генерувати відповідний Python-код, який можна використовувати у вашому додатку.

Pyuic5 читає файл .ui, створений у Qt Designer, і створює файл .py, який містить клас Python, що представляє інтерфейс користувача. Цей згенерований код може бути імпортований у додатку і використаний для відображення та взаємодії з елементами інтерфейсу.

Використання Pyuic5 дозволяє значно спростити розробку графічного інтерфейсу у PyQt5, оскільки можна сконцентруватись на візуальному проектуванні у Qt Designer, а не писанні вручну великої кількості коду. Pyuic5 виконує важливу роль у швидкому і ефективному розвитку програм на основі PyQt5.

Для виконання перетворення потрібно використовувати команду pyuic5 у командному рядку. Зазвичай, команда виглядає наступним чином: pyuic5 input.ui -o output.py, де input.ui - це шлях до файлу .ui, а output.py - це шлях до файлу .py, в який буде збережено згенерований код.

У деяких випадках можна використовувати окремий виконавчий файл для pyuic5 замість прямого виклику команди pyuic5. Цей виконавчий файл може знаходитись у папці PyQt5 або міститись у Python-середовищі. У такому випадку потрібно вказати шлях до виконавчого файлу pyuic5 у команді, наприклад: python path/to/pyuic5 input.ui -o output.py.

3.3. Написання логіки додатку

Натискання на кнопку View буде викликати метод change, який відповідає за зміну видимості елементів у графічному інтерфейсі. Якщо self.swap має значення True, то елемент textEdit_2 піднімається на передній план, в іншому випадку - елемент treeWidget піднімається на передній план. Після цього, значення self.swap змінюється на протилежне. По стандарту значення swap відповідає False, відповідно спочатку на екрані відображається textEdit, тобто текстове поле яке буде відображати рядок JSON після парсингу. Для переносу елементу вище всіх інших використовується метод .raise_().

```
def change(self):
    if self.swap:
        self.textEdit_2.raise_()
    else:
        self.treeWidget.raise_()
    self.swap = not self.swap
```

Рис. 3.12. Метод change()

Після цього, потрібна функція, яка після натискання на кнопку Parse отримує JSON-рядок з першого текстового поля textEdit, розпаковує його у JSON-об'єкт за допомогою json.loads(). Далі, використовуючи json.dumps() та параметр indent=4, вона форматує JSON-об'єкт у зрозумілий формат з використанням відступів. Отриманий відформатований JSON-рядок вставляється у друге текстове поле textEdit_2. Далі, викликається функція, яка вставляє JSON-об'єкт у QTreeWidget для відображення даних у деревоподібній структурі.

```python
def format_json(self):
    # Отримання JSON-рядка з першого текстового поля
    json_str = self.textEdit.toPlainText()
    # Розпакування JSON-рядка
    json_obj = json.loads(json_str)
    # Форматування JSON-об'єкта у зрозумілий формат з використанням відступів
    formatted_json = json.dumps(json_obj, indent=4)
    # Очищення та вставлення відформатованого JSON-об'єкта у друге текстове поле
    self.textEdit_2.clear()
    self.textEdit_2.insertPlainText(formatted_json)
    # Очищення QTreeWidget перед вставкою нових даних
    self.treeWidget.clear()
    # Вставка кореневого елементу в QTreeWidget
    self.insert_json_item(self.treeWidget, json_obj, expanded=True)
```

Рис. 3.13. Метод format_json()

Функція insert_json_item виконує рекурсивне додавання елементів у QTreeWidget. Рекурсивне додавання елементів у QTreeWidget означає, що процес додавання елементів виконується вкладено, тобто одна функція викликає себе саму для обробки піделементів.

У випадку з QTreeWidget, рекурсивне додавання елементів означає, що функція, викликаючись для певного елементу, перевіряє його тип. Якщо це словник, вона створює вузол з назвою словника та розглядає всі його елементи, викликаючи саму себе для кожного з них. Те саме відбувається із списками - створюється вузол з назвою списку, а елементи списку обробляються рекурсивно.

Рекурсивний підхід дозволяє обробляти складні структури даних, які можуть містити вкладені словники або списки. Він дозволяє проходити по кожному елементу і виконувати необхідні дії, а також додавати його до QTreeWidget як підвузол.

Рекурсивне додавання елементів у QTreeWidget зазвичай використовується для відображення деревоподібних структур даних, де кожен елемент може мати довільну кількість піделементів. Вона дозволяє створити багаторівневу ієрархію елементів з урахуванням їх взаємозв'язків та вкладеності [4-6].

ВИСНОВКИ ДО РОЗДІЛУ 3

У процесі розробки було використано такі елементи інтерфейсу, як TextEdit для введення та відображення JSON-рядка, TreeWidget для відображення структурованих даних у вигляді дерева, а також розміщення елементів у відповідних макетах (Layouts). Для зручного керування програмою були використані Push Buttons для виконання певних операцій, таких як форматування та відображення даних.

Програма реалізує алгоритм роботи, який включає такі кроки як тримання JSON-рядка з TextEdit, розпакування JSON-рядка у JSON-об'єкт за допомогою відповідної бібліотеки, форматування JSON-об'єкту з використанням відступів. відображення відформатованого JSON-рядка у TextEdit_2, очищення та вставлення даних у TreeWidget, рекурсивне додавання елементів у QTreeWidget для відображення структури JSON даних.

Ця реалізація дозволила створити функціональний JSON парсер, який спрощує роботу з базою даних та обробку JSON інформації. За допомогою наданого інтерфейсу користувач може ввести JSON-рядок, візуалізувати його структуру та здійснити необхідні операції над даними. Програма є потужним інструментом для роботи з JSON форматом та сприяє ефективному управлінню даними у БД.

РОЗДІЛ 4
ПІДКЛЮЧЕННЯ ДО БАЗИ ДАНИХ, СТИЛІЗАЦІЯ, РЕАЛІЗАЦІЯ

4.1. Підключення проекту до бази даних

База даних - це організований набір структурованої інформації або даних, які зазвичай зберігаються в електронному вигляді в комп'ютерній системі. Зазвичай базою даних керує система управління базами даних (СКБД). Разом дані та СУБД, а також додатки, які з ними пов'язані, називаються системою баз даних, яку часто скорочують до просто бази даних.

Дані в найпоширеніших типах баз даних, що використовуються сьогодні, зазвичай моделюються у вигляді рядків і стовпців у серії таблиць, щоб зробити обробку та запити до даних ефективними. Тоді до даних можна легко отримати доступ, керувати ними, змінювати, оновлювати, контролювати та організовувати. Більшість баз даних використовують мову структурованих запитів (SQL) для запису і запитів до даних.

Формат JSON використовується для впорядкованого зберігання даних у процесі їх обміну між веб-браузером (або клієнтською частиною програми) та сервером (або між різними серверами). Більше того, завдяки текстовому вигляду рядка дані JSON можна легко передавати через будь-які інші канали обміну інформацією в інтернеті. І, звісно, без нього неможливо обійтися в сучасних великих базах даних, де потрібно зберігати масу інформації.

В даному прикладі буде розглянуто підключення проекту Python до бази на основі PostgreSQL. Підключення до бази може бути реалізоване за допомогою бібліотеки psycopg2. Щоб її встановити, потрібно відкрити командний рядок і прописати «pip install psycopg2».

Далі, потрібно налаштувати параметри підключення до бази даних PostgreSQL. Ці параметри включають ім'я хоста (наприклад, "localhost"), номер порту (за замовчуванням 5432), ім'я бази даних, ім'я користувача та пароль.

```
def display_data():
    # Підключення до бази даних
    conn = psycopg2.connect(
        host="localhost",
        port="5432",
        database="northwind",
        user="postgres",
        password="1111"
    )
```

Рис. 4.1. Приклад коду для підключення до БД

Для встановлення з'єднання з базою даних використовується psycopg2.connect(). Якщо з’єднання успішне, то змінна conn буде містити об'єкт з'єднання з базою даних PostgreSQL.

Після успішного підключення можуf виконувати операції з базою даних, такі як виконання SQL-запитів, створення, зміна або видалення таблиць, вставка, оновлення або видалення даних тощо. Наприклад можна використовувати метод execute() для виконання SQL-запитів.

```
cursor = conn.cursor()  # Створення курсора
cursor.execute("SELECT * FROM your_table")  # Виконання SQL-запиту
result = cursor.fetchall()  # Отримання результатів запиту
cursor.close()  # Закриття курсора
```

Рис. 4.2. Приклад використання execute()

Після закінчення роботи з базою даних важливо закрити курсор та з'єднання. Використовується метод close() для закриття курсора та закриття з'єднання [11].

4.2. Стилізація проекту за допомогою CSS

CSS (Cascading Style Sheets) - це мова опису стилів, яка використовується для оформлення веб-сторінок. Вона визначає зовнішній вигляд елементів HTML-документа, таких як кольори, шрифти, розміри, відступи, фони та інші візуальні властивості.

У Qt Designer CSS використовується для оформлення та налаштування зовнішнього вигляду елементів інтерфейсу. Використання CSS у Qt Designer дозволяє змінювати стилі елементів, задавати їх кольори, фони, шрифти, вирівнювання та багато іншого.

CSS дозволяє розділити логіку стилізації від розмітки та функціональності, що спрощує редагування та зміну зовнішнього вигляду елементів без необхідності втручатися в код програми. Це дозволяє швидко та зручно змінювати стиль інтерфейсу, а також забезпечує більшу гнучкість і переносимість дизайну між різними платформами та пристроями [8].

CSS можна підключити за допомогою .setStyleSheet(). Так як користувацький інтерфейс програми ще виглядає дуже сіро і непривабливо, можна почати зі зміни заднього фону, для цього наприклад підійде background-color. Щоб зробити градієнтну заливку можна використовувати background-color: qlineargradient(spread:pad, x1:0, y1:0, x2:1, y2:0, stop:0 rgb(color1), stop:1 rgb(color2));

Оскільки за замовчуванням стиль головного вікна поширюється на всі віджети, то будемо мати наступний вигляд:

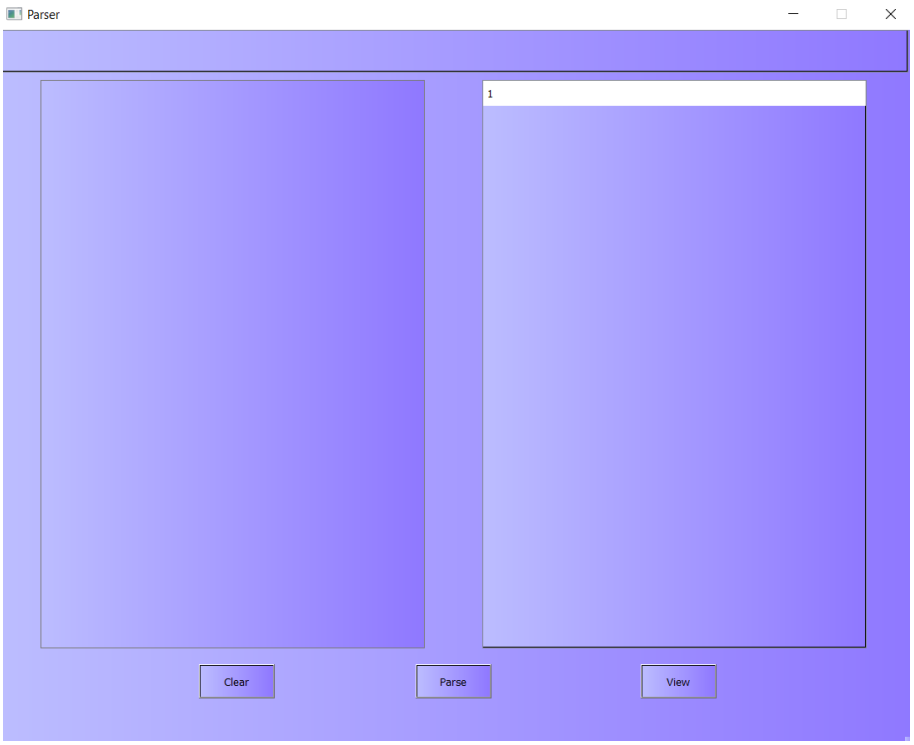

Рис. 4.3. Зміна заднього фон у

Далі можна стилізувати кнопки, дати їм однотонний колір заокруглеггя по кутам, та зробити красивіший текст.

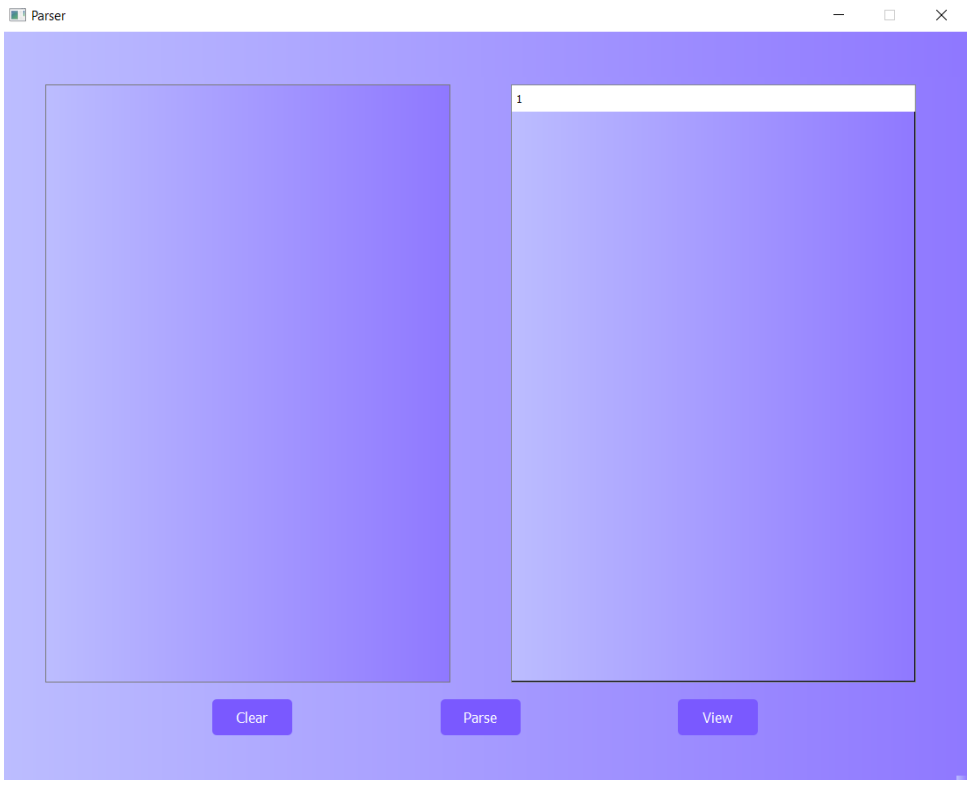

Рис. 4.4. Стилізація кнопок

TextEdit та treeWidget можна зробити з білим заднім фоном, заокруглити кути, щоб відповідало кнопкам і здавалось більш сучасним та стилізувати текст, його шрифат, розмір, розташування. В кінечному вигляді програма буде виглядати наступним чином:

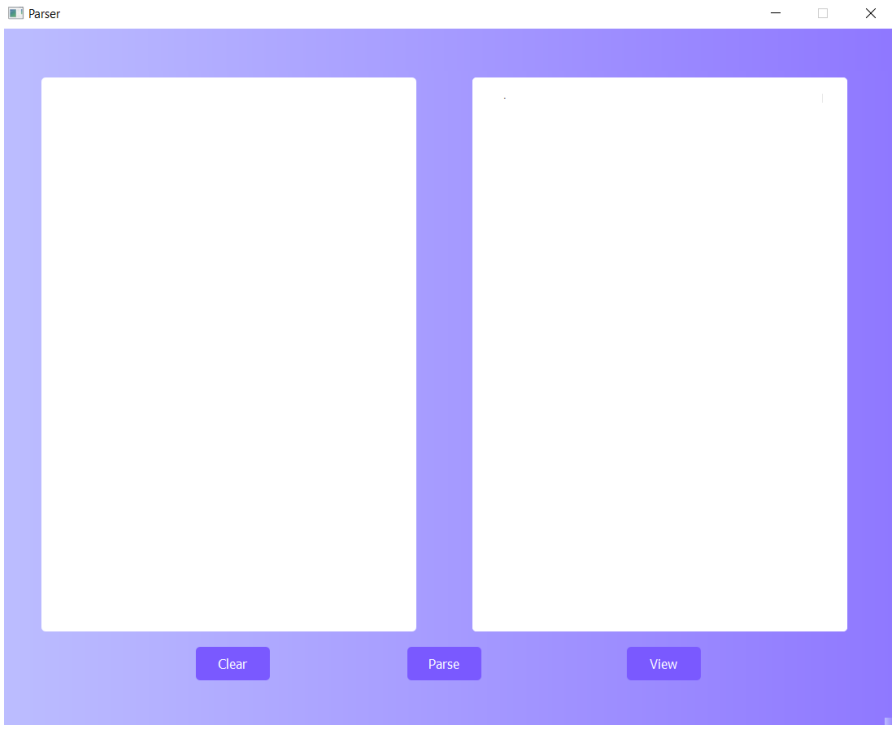

Рис. 4.5. Кінцевий вигляд програми

4.3. Реалізація програми в .exe форматі

Auto-py-to-exe (Auto PY to EXE) - це інструмент, який дозволяє легко перетворювати програми Python у виконувані файли .exe. Він надає графічний інтерфейс користувача, який спрощує процес створення виконуваного файлу з вашого Python-коду.

Auto-py-to-exe пропонує широкі можливості налаштування, що дозволяє вказувати параметри, такі як вхідний файл Python, залежності, іконку, назву програми, параметри запуску та багато іншого. Також можна включити залежності в проект, щоб мати один виконуваний файл, який містить всі необхідні компоненти [7].

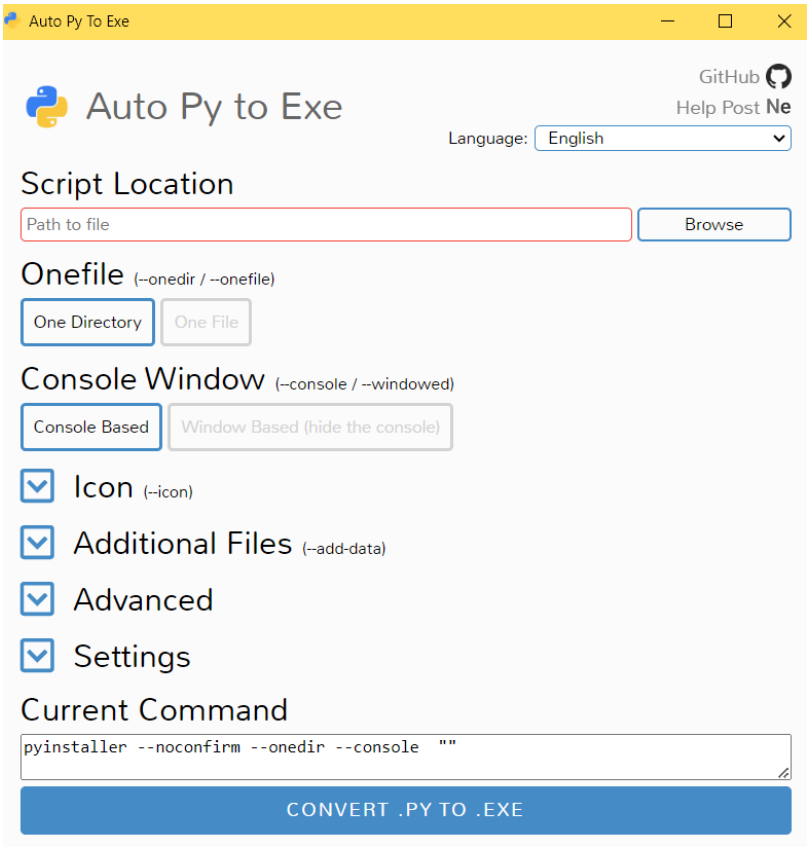

Рис. 4.6. Інтерфейс Auto-py-to-exe

Для реалізації програми Python у вигляді виконуваного файлу .exe за допомогою auto-py-to-exe буде потрібно:

1. Спочатку потрібно встановити auto-py-to-exe на комп'ютері. Це можна зробити за допомогою пакетного менеджера pip, виконавши команду pip install auto-py-to-exe в командному рядку.
2. Після встановлення можна запустити auto-py-to-exe, щоб створити конфігураційний файл для програми. У цьому конфігураційному файлі можна вказати різні налаштування, такі як вхідний файл Python, вимоги до залежностей, тип виконуваного файлу, інклюди та інше.
3. Після створення конфігураційного файлу потрібно налаштувати параметри відповідно до проекту. Наприклад, вказати, щоб був створений один .exe файл або набір файлів, вибрати іконку, задати

назву програми тощо. Також можна включити різні залежності, які програма вимагає для правильної роботи.

4. Після налаштування параметрів потрібно запустити процес збірки програми. Auto-py-to-exe виконає конвертацію програми Python у виконуваний файл .exe згідно з вказаними налаштуваннями. Після завершення процесу створюється готова програма.

Після отримання виконуваного файлу .exe його можна протестувати, переконавшись, що програма працює належним чином. Якщо все в порядку, цей файл можна використовувати на різних комп'ютерах без необхідності наявності Python.

Дуже важливо протестувати програму на різних пристроях, та різних розширеннях, так як наприклад вікно з розмірами 1100 x 850 не буде коректно працювати на пристроях, де розширення дисплея дорівнюватиме 1400 x 1050 або менше. Тож потрібно встановити таке стартове розширення, яке буде відповідати всім нормам, а при бажанні, користувач сам зможе розширити головне вікно застосунку.

ВИСНОВКИ ДО РОЗДІЛУ 4

В результаті було успішно реалізовано підключення бази даних PostgreSQL до проекту на мові програмування Python. Застосування бібліотеки psycopg2 дозволило забезпечити ефективне управління даними, здійснюючи взаємодію з базою даних, виконуючи SQL-запити та маніпулюючи збереженими даними.

Шляхом використання CSS, була виконана стилізація проекту, надаючи йому привабливий та сучасний зовнішній вигляд. Використання CSS дозволило налаштувати колірну гаму, шрифти, розміри та інші властивості елементів інтерфейсу, забезпечуючи їх консистентний та привертаючий увагу вигляд.

Завершуючим етапом стало перетворення розробленого проекту на виконуваний файл .exe з використанням інструменту auto-py-to-exe. Це дозволило зручно розповсюджувати та запускати програму на різних пристроях без необхідності встановлення середовища Python та його залежностей. Такий формат файлу забезпечує зручну розповсюдження та використання проекту, що є важливим фактором для його доступності та практичного використанн

ВИСНОВКИ

У сучасному світі обробка та аналіз даних стають все більш важливими завданнями, що передують успішному розвитку технологій та бізнесу. Один із найпоширеніших форматів для обміну та зберігання даних - JSON, знайшов широке застосування в сфері програмування та баз даних. JSON має просту та зручну структуру, що дозволяє легко обробляти дані, та його популярність зростає з кожним днем.

Однак, для ефективної роботи з даними у форматі JSON потрібен потужний інструмент, який зможе зчитувати та обробляти дані відповідно до потреб користувача. Таким інструментом є JSON-parser, програмне забезпечення, яке дозволяє перетворювати дані у форматі JSON у зручний та читабельний вигляд для подальшої обробки.

Після ретельного порівняльного аналізу трьох середовищ розробки для Python - Python IDLE, PyCharm і Visual Studio Code, я прийшов до висновку, що PyCharm є найкращим варіантом в моєму випадку. Щодо бібліотек, то були обрані бібліотеки PyQt5, Qt, json, pyjq.

У процесі розробки було використано такі елементи інтерфейсу, як TextEdit для введення та відображення JSON-рядка, TreeWidget для відображення структурованих даних у вигляді дерева, а також розміщення елементів у відповідних макетах (Layouts). Для зручного керування програмою були використані Push Buttons для виконання певних операцій, таких як форматування та відображення даних.

Програма реалізує алгоритм роботи, який включає такі кроки як тримання JSON-рядка з TextEdit, розпакування JSON-рядка у JSON-об'єкт за допомогою відповідної бібліотеки, форматування JSON-об'єкту з використанням відступів. відображення відформатованого JSON-рядка у

TextEdit_2, очищення та вставлення даних у TreeWidget, рекурсивне додавання елементів у QTreeWidget для відображення структури JSON даних.

Ця реалізація дозволила створити функціональний JSON парсер, який спрощує роботу з базою даних та обробку JSON інформації. За допомогою наданого інтерфейсу користувач може ввести JSON-рядок, візуалізувати його структуру та здійснити необхідні операції над даними. Програма є потужним інструментом для роботи з JSON форматом та сприяє ефективному управлінню даними у базі даних.

Було успішно реалізовано підключення бази даних PostgreSQL до проекту на мові програмування Python. Застосування бібліотеки psycopg2 дозволило забезпечити ефективне управління даними, здійснюючи взаємодію з базою даних, виконуючи SQL-запити та маніпулюючи збереженими даними.

Завершуючим етапом стало перетворення розробленого проекту на виконуваний файл .exe з використанням інструменту auto-py-to-exe. Це дозволило зручно розповсюджувати та запускати програму на різних пристроях без необхідності встановлення середовища Python та його залежностей. Такий формат файлу забезпечує зручну розповсюдження та використання проекту, що є важливим фактором для його доступності та практичного використання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. «JavaScript: The Good Parts» Douglas Crockford 172 с. - видавництво: O'Reilly Media - режим доступу до ресурсу: https://cdn.tc-library.org/Rhizr/Files/daaz74mzphKfnHsen/files/JavaScript-%20The%20Good%20Parts.pdf

2. "Learning JavaScript Data Structures and Algorithms" Lois Truver 426 с. /режим доступу: https://www.packtpub.com/product/learning-javascript-data-structures-and-algorithms-third-edition/9781788623872

3. "JavaScript and JSON Essentials" by Alex R. Young / режим доступу: https://www.packtpub.com/product/javascript-and-json-essentials/9781783286034

4. "Python Crash Course" by Eric Matthes

5. "Python for Data Analysis" by Wes McKinney

6. Python IDLE [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/library/idle.html

7. Auto-py-to-exe [Електронний ресурс] – режим доступу до ресурсу: https://stackoverflow.com/questions/22266802/how-to-convert-a-ui-file-to-py-file

8. CSS [Електронний ресурс] – Режим доступу до ресурсу: https://www.bigcommerce.com/ecommerce-answers/what-css-and-why-it-important/

9. PyCharm [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com/ru-ru/pycharm/features/

10. Бібліотек PyQt5 [Електронний ресурс] – Режим доступу до ресурсу: https://python-scripts.com/pyqt5

11. Бібліотека Psycopg2 [Електронний ресурс] – Режим доступу до ресурсу: https://pypi.org/project/psycopg2/